Bridging Perception and Reasoning in Table Understanding: The Path from Recognition to Trustworthy and Explainable AI

A thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Engineering

by

Sachin Raja
2020801001
sachin.raja@research.iiit.ac.in

Advisor: Dr. CV Jawahar

International Institute of Information Technology Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA

April 2026

International Institute of Information Technology Hyderabad
Hyderabad, India

CERTIFICATE

This is to certify that work presented in this thesis proposal titled ***Bridging Perception and Reasoning in Table Understanding: The Path from Recognition to Trustworthy and Explainable AI*** by *Sachin Raja* has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Advisor: Dr. CV Jawahar

To my Late Maternal and Paternal Grand Parents

Sh. Chandrasen Agrawal & Smt. Kailash Devi,

Dr. OP Gupta & Smt. Kaushalya Devi,

my parents *Dr. Suman & Dr. Deepak Raja*,

my wife *Ayushi Gupta*,

and All my *Teachers and Mentors*

Acknowledgments

I would like to express my deepest gratitude to my advisor, **Prof. C. V. Jawahar**, for his invaluable guidance, encouragement, and unwavering support throughout my Ph.D. journey. His vision, clarity of thought, and constant pursuit of excellence have profoundly shaped my research philosophy and academic development. I am immensely grateful for the opportunity to learn from his expertise and for the intellectual freedom he entrusted me with in pursuing ambitious research directions.

My sincere thanks go to **Dr. Ajoy Mondal**, with whom I had the privilege of working closely as a collaborator and mentor. His technical acumen, patience, and detailed insights have been instrumental in shaping every aspect of my research—from conceptualization to implementation and evaluation. His mentorship not only strengthened my understanding of experimental design and scientific rigor but also deeply influenced the way I approach research problems.

I would also like to thank **Dr. Alok Aggarwal** for his continued mentorship during my time at Scry AI. His guidance helped me strengthen my analytical foundations and instilled in me the confidence to take on challenging research problems. The experience of working under his supervision laid a strong technical groundwork that proved invaluable in my doctoral research.

I am forever indebted to my **family** for their unconditional love, patience, and belief in me. Their sacrifices and encouragement have been my greatest source of motivation. I owe special gratitude to my **wife** and my **grandfather**, whose steadfast support, understanding, and resilience carried me through the highs and lows of the Ph.D. journey. Their unwavering faith in me has been the emotional anchor behind this work.

Finally, I wish to thank all my **colleagues at the CVIT Lab, IIIT-Hyderabad**, for their camaraderie, insightful discussions, and collaborative spirit. The vibrant and intellectually stimulating environment of the lab has played a crucial role in enriching both my research and personal growth.

This thesis stands as a culmination of the collective guidance, support, and kindness of all these individuals, to whom I owe my deepest appreciation.

Abstract

This doctoral research presents a comprehensive and multi-faceted investigation into automated table understanding, arguing that robust and reliable solutions demand an approach that evolves from foundational structural parsing to address the pressing real-world requirements of data privacy and auditable reasoning. Tables are information-rich, structured objects that serve as a cornerstone for conveying complex data, yet their automated parsing is a formidable, long-standing challenge in document intelligence. The core of this challenge lies in Table Structure Recognition (TSR), the process of transforming a table image into a structured, machine-readable format. The difficulty is rooted in the immense visual diversity of tables, with complexities such as spanning cells, multi-line text, and the absence of ruling lines often causing traditional and early deep learning methods to fail. This body of work charts a clear research trajectory that begins with the development of a novel framework for TSR, TabStruct-Net, and progressively refines this methodology to handle increasing visual complexity. The research then pivots to address critical non-functional requirements, pioneering TabGuard, a novel framework for privacy-preserving TSR, and finally extends its scope from structure to trusted reasoning by introducing EviFiVQA, a benchmark for financial Visual Question Answering (VQA) that establishes evidence localization as a core tenet of auditable AI. This journey from pixels to privacy and proof marks a significant contribution to the field.

The core methodology advanced throughout this research is anchored in a powerful two-step paradigm that mirrors human cognitive processes: a top-down decomposition followed by a bottom-up reconstruction. In the top-down phase, the table image is decomposed into its fundamental constituent parts—the individual table cells—through an object detection model. In the bottom-up phase, the global table structure is reconstructed by learning the spatial and logical associations between the detected cells. A cornerstone of this research is the novel insight that TSR performance can be dramatically improved by encoding human intuition about table structure directly into the learning objective. This was achieved through a series of innovative, cognitive-inspired loss functions that act as structural regularizers, including an Alignment Loss to enforce a grid-like structure, a Continuity Loss to ensure adjacent cell boundaries are contiguous, and an Overlapping Loss to penalize spatial conflicts. This approach is marked by a clear architectural evolution, beginning with TabStruct-Net, which combined a modified Mask R-CNN with a Dynamic Graph Convolutional Neural Network, and culminating in TabStruct-Net V2, which introduced a Hierarchical Local-Attention Vision Transformer (HLVIT) backbone and a highly efficient self-attention layer to achieve state-of-the-art performance and scalability.

Building upon the foundation of accurate structural recognition, the research pivots to address a critical real-world barrier to the adoption of document AI: data privacy. The research introduces TabGuard, a pioneering framework that enables privacy-preserving TSR through a novel client-server architecture. In this model, the client performs content masking locally, blacking out all text regions before sending only the anonymized image to the server for structure recognition. This effectively decouples structural understanding from content recognition, ensuring sensitive data is never exposed. The key technical innovation that makes this possible is the Table Grid Approximator (TGA), a server-side module that analyzes the spatial distribution of masked text contours to infer a coarse structural grid. This grid serves as a powerful inductive bias, allowing for the dynamic generation of high-quality, layout-specific anchor boxes that enable the detection network to converge accurately even without access to text content. The success of TabGuard provides compelling evidence that a table's structure can be robustly inferred purely from its spatial layout cues, independent of its content.

The final thrust of this research extends beyond structural parsing to the higher-level goal of building trustworthy AI systems for semantic document understanding, explored in the high-stakes domain of financial analysis. Identifying a critical gap in existing VQA benchmarks, which lack support for the complex numerical and hierarchical reasoning required for financial documents, the EviFiVQA benchmark was created. Its most significant contribution is the requirement of evidence localization: for each question, a model must produce not only the correct answer but also the bounding boxes of all table cells used to compute it. This feature transforms VQA from a black-box task into a transparent and interpretable one, providing a direct mechanism for auditability by allowing an analyst to trace the model's conclusions back to the source data. This work on evidence-grounded VQA is the logical culmination of the foundational research on high-precision TSR, as the ability to demand that a model "show its work" is entirely predicated on a system that can first perceive and delineate cells with high fidelity. Collectively, this body of work pushes the field of document intelligence beyond data extraction towards the creation of systems that are simultaneously accurate, robust, secure, and transparent, with future work poised to extend these principles to other structured document objects and deploy them in production-level systems for financial, legal, and medical document processing.

Keywords: Document Image Analysis, Table Structure Recognition (TSR), Top-Down and Bottom-Up Cues, Cognitive-Inspired Constraints, Privacy-Preserving TSR, Content-Masked Image Analysis, Evidence-Grounded Reasoning, Financial Visual Question Answering, Auditable AI, Complex Table Layouts

Contents

List of Figures

List of Tables

List of Related Publications

[P1] **S. Raja**, A. Mondal, C.V. Jawahar. *"From pixels to tables: reconstructing complex tables from document images."* **International Journal on Document Analysis and Recognition (IJDAR) (2025): 1-16.**

[P2] **S. Raja**, A. Mondal, C.V. Jawahar. *"EviFiVQA: A Benchmark for Evidence-Grounded Multi-hop Reasoning in Financial VQA."* **International Conference on Document Analysis and Recognition. Cham: Springer Nature Switzerland, 2025**

[P3] **S. Raja**, A. Mondal, C.V. Jawahar. *"Treading Towards Privacy Preserving Table Structure Recognition."* **IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2025.**

[P4] **S. Raja**, A. Mondal, C.V. Jawahar. *"ICDAR 2023 Competition on Visual Question Answering on Business Document Images."* **17th International Conference on Document Analysis and Recognition (ICDAR), 2023, pages 454-470.**

[P5] **S. Raja**, A. Mondal, C.V. Jawahar. *"Visual Understanding of Complex Table Structures from Document Images."* **IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022**, pages 2299-2308.l

[P6] **S. Raja**, A. Mondal, C.V. Jawahar. *"Table Structure Recognition using Top-Down and Bottom-Up Cues."* **European Conference on Computer Vision (ECCV), Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII, 16.** *(Citations: 118)*

Related co-author publications:

[P7] H.S. Bhatt, S. Ramakrishnan, **S. Raja**, C.V. Jawahar. *"Unlocking the Potential of Unstructured Data in Business Documents Through Document Intelligence."* **11th ACM India Joint International Conference on Data Science and Management of Data (CODS-COMAD), 2024.**

[P8] P. Goyal, **S. Raja**, D. Huang, A. Mondal, C.V. Jawahar. *"Graph Representation Ensemble Learning."* **IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2020.**

Chapter 1

Introduction

Tables are ubiquitous information-rich structures that appear across diverse document types, from financial statements and scientific publications to business reports and regulatory filings. The ability to automatically understand and process tabular information from document images represents a critical capability in the broader landscape of intelligent document processing systems [1], [2]. As organizations worldwide grapple with exponentially growing volumes of document-based data, the challenge of extracting structured information from tables has evolved from a niche research problem into a fundamental requirement for modern document intelligence systems.

The proliferation of digital documents in multiple formats—ranging from scanned paper documents to born-digital PDFs and web-based tables—has created an urgent need for robust table understanding systems that can operate across diverse visual characteristics, layout complexities, and domain-specific conventions [3], [4]. While significant progress has been made in localizing tables as graphical objects within document images [5]–[7], the more challenging task of recognizing their internal structure—decomposing tables into constituent cells and understanding their hierarchical relationships—remains an active area of research with substantial room for improvement.

1.1 Intelligent Document Processing: The Foundation

Intelligent Document Processing (IDP) encompasses a broad spectrum of technologies designed to automatically extract, classify, and understand information from various document types. Traditional document processing pipelines relied heavily on rule-based systems and manual data entry, resulting in inefficient workflows that were both time-consuming and error-prone [8], [9]. The advent of deep learning has fundamentally transformed this landscape, enabling end-to-end trainable systems capable of learning complex visual and linguistic patterns directly from data [1], [2].

Modern IDP systems integrate multiple cognitive tasks including document classification, layout analysis, object detection, optical character recognition (OCR), and semantic understanding [10], [11].

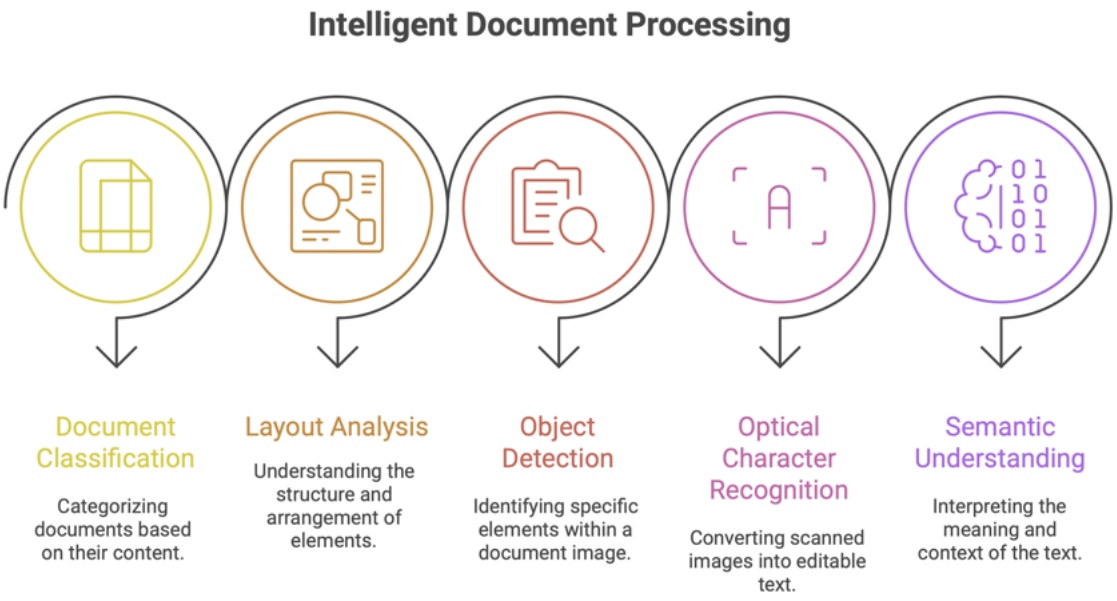


Figure 1.1: Some pivotal components of Intelligent Document Processing.

Within this ecosystem, table understanding represents a particularly challenging component due to the dense information encoding, structural variations, and the need to preserve both spatial layout and semantic relationships [3]. Tables serve as compressed representations of multi-dimensional data, where rows and columns organize information according to implicit or explicit hierarchical schemas. The challenge lies not merely in recognizing the presence of tabular structures but in accurately reconstructing their logical organization while preserving the semantic associations between cells [12].

The importance of robust table processing extends across numerous application domains. In financial services, automated extraction of data from balance sheets, income statements, and cash flow statements enables rapid credit risk assessment and regulatory compliance [13]. Healthcare organizations require accurate extraction of patient data from clinical forms and lab reports [3]. Legal document analysis, academic literature mining, government record digitization, and business intelligence all depend critically on the ability to accurately parse tabular information from diverse document sources.

1.2 Tables in Document Images: A Multifaceted Challenge

Tables manifest in document images through multiple channels, each presenting distinct characteristics and challenges. Understanding these diverse sources is essential for developing generalizable

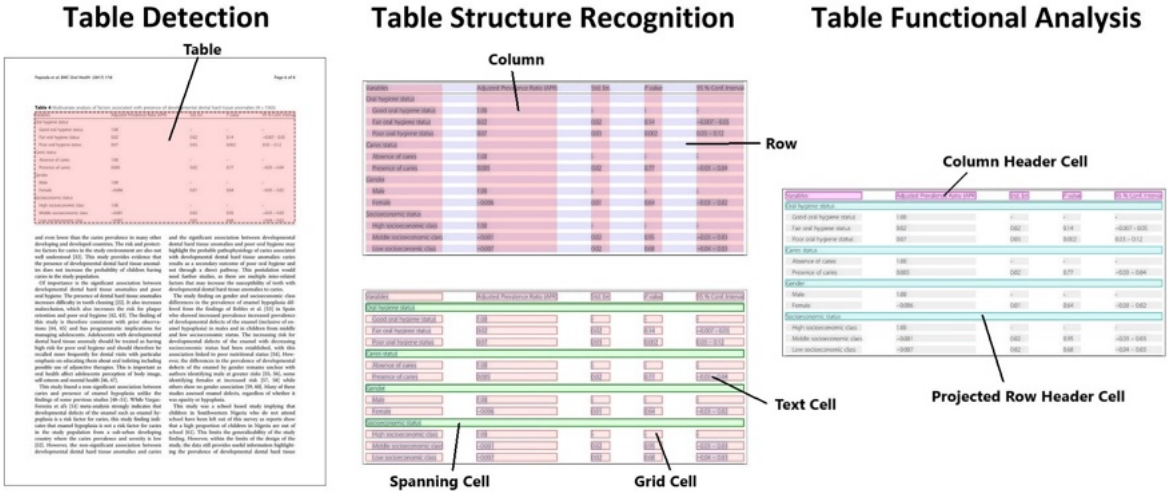


Figure 1.2: Outline of Table Understanding.

table understanding systems that can operate robustly across different document types and acquisition modalities [3], [4].

1.2.1 Sources of Digital Tables

Web Tables: Tables extracted from HTML documents represent one of the earliest sources of large-scale tabular data [14], [15]. Web tables benefit from the presence of structural markup that explicitly defines rows, columns, headers, and cell spanning information. However, web tables exhibit enormous diversity in formatting conventions, CSS styling, and semantic organization. Many web tables lack consistent header structures or use unconventional layouts that complicate automated processing [16], [17].

Born-Digital PDF Documents: PDF documents generated directly from digital sources such as word processors, spreadsheet applications, or typesetting systems contain embedded metadata that can facilitate table extraction [18]. However, PDF format specifications do not mandate the preservation of logical table structure. Text and graphical elements are often rendered as independent objects without explicit structural relationships, requiring sophisticated reconstruction algorithms to recover the original tabular organization [8], [19].

Scanned Document Images: A substantial portion of historical documents, business records, and archival materials exist only as scanned images. These documents present the most challenging scenario for table understanding, as they lack any embedded structural information [1], [2]. Scanned documents may suffer from image quality degradations including noise, blur, skew, perspective distortion, and uneven illumination. Furthermore, scanning artifacts such as ink bleeding, faded text, and paper deteri-

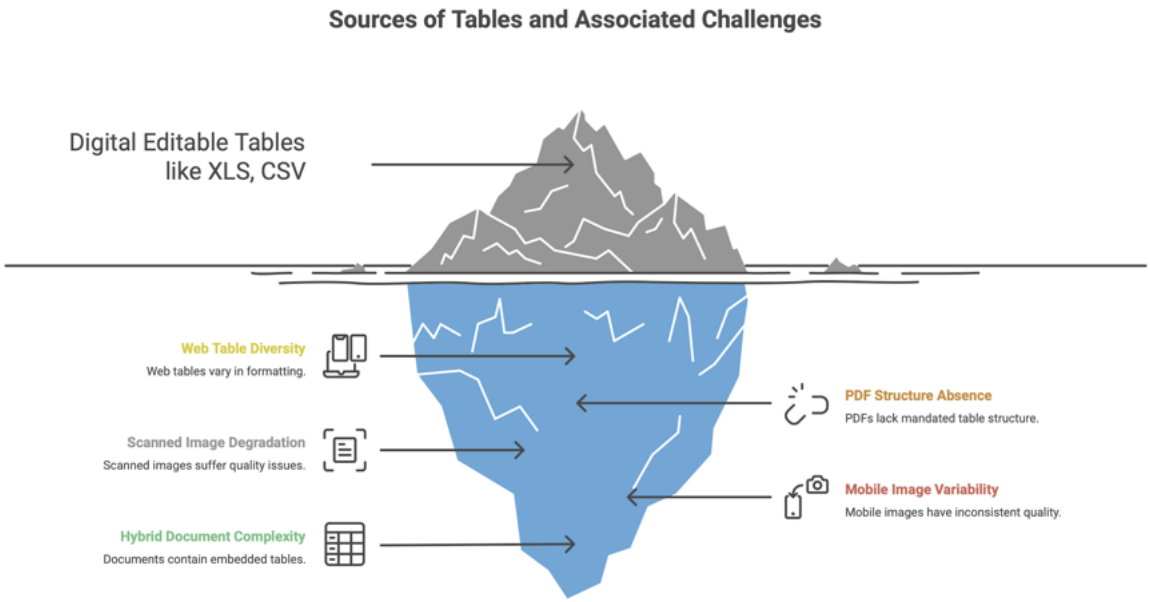


Figure 1.3: Sources of Tables and Associated Challenges.

oration compound the difficulty of accurate table reconstruction [20], [21].

Mobile-Captured Document Images: With the proliferation of smartphones and mobile document scanning applications, an increasing volume of document images are captured using mobile cameras [22]. These images exhibit additional challenges including perspective distortion, non-uniform lighting, shadows, and motion blur. The casual capture conditions typical of mobile photography result in highly variable image quality that challenges robust table understanding [22].

Hybrid and Multi-Page Documents: Real-world documents frequently contain tables embedded within rich textual content, accompanied by figures, charts, and other graphical elements [23]. Multi-page financial reports may split tables across page boundaries, requiring sophisticated document-level reasoning to reconstruct complete tabular structures. Furthermore, some documents contain nested or hierarchical table structures that challenge conventional two-dimensional grid assumptions [24].

1.2.2 Visual and Structural Complexity

The visual presentation of tables spans a vast spectrum of complexity [1]. Simple grid-structured tables with explicit ruling lines and uniform cell sizes represent the most straightforward case. However,

Figure 1.4: Variations include scanned document images, food ingredient labels, Excel tables, and invoice tables.

real-world tables frequently deviate substantially from this idealized format [9].

Tables may lack horizontal or vertical separator lines entirely, relying solely on whitespace and text alignment to convey structure [25]. Cells within a single table can exhibit dramatic variations in size, shape, and aspect ratios. Multi-row and multi-column spanning cells are common, particularly in header regions and when representing hierarchical categorizations [1], [2]. Empty cells, which carry important semantic meaning by indicating null values or inapplicable categories, pose significant detection challenges as they contain no visual features beyond boundaries [2].

Dense tables with numerous rows and columns compressed into limited page space present particularly acute challenges [26]. Cell content may include multi-line text, mathematical formulas, monetary values with special formatting, percentages, dates in varied formats, and even nested sub-tables [24]. Text within cells may be left-aligned, right-aligned, centered, or justified, with varying fonts, sizes, and

styles used to convey semantic distinctions [9].

Effective table understanding begins with extracting discriminative visual features that capture both local cell characteristics and global table structure [27], [28]. Convolutional neural networks have proven effective for learning hierarchical visual representations from document images [29], [30]. However, tables present unique challenges for feature extraction. The extreme aspect ratios and scale variations of table cells challenge standard object detection architectures that assume roughly square or moderately rectangular objects [1].

Furthermore, cells within tables are not independent objects but rather components of a coherent structure constrained by alignment, continuity, and non-overlapping properties [1], [2]. Standard object detection approaches that treat cells as independent instances fail to exploit these structural constraints, resulting in misaligned detections and overlapping cell boundaries. The presence of empty cells, which lack distinctive visual features, requires reasoning about negative space and inferring structure from surrounding context [2].

1.3 Problem Definition and Formulation

Table understanding is a complex task that involves interpreting and extracting information from tables within documents, converting them into machine-readable formats, and enabling advanced interactions such as querying and reasoning [1], [2]. Two critical components of table understanding are table structure recognition and visual question answering on tables once the tables are located within the documents. Achieving a holistic view of table understanding involves dividing the process into three interrelated sub-problems — detection of tables within the document, structure recognition through cell detection and localization, and semantic understanding of its content.

1.3.1 Table Detection

Table detection from document images is a crucial task in document image analysis that aims to automatically locate and delineate tabular regions within scanned or digitally rendered documents such as invoices, financial statements, forms, and scientific articles. Tables often encode structured information in a compact form, making their detection an essential first step for downstream tasks like table structure recognition, table content extraction, and document understanding. However, this problem is challenging due to the diverse appearances of tables—ranging from ruled to unruled layouts, variations in font, scale, and alignment, as well as complex page backgrounds and overlapping non-table elements like text blocks, figures, or graphics.

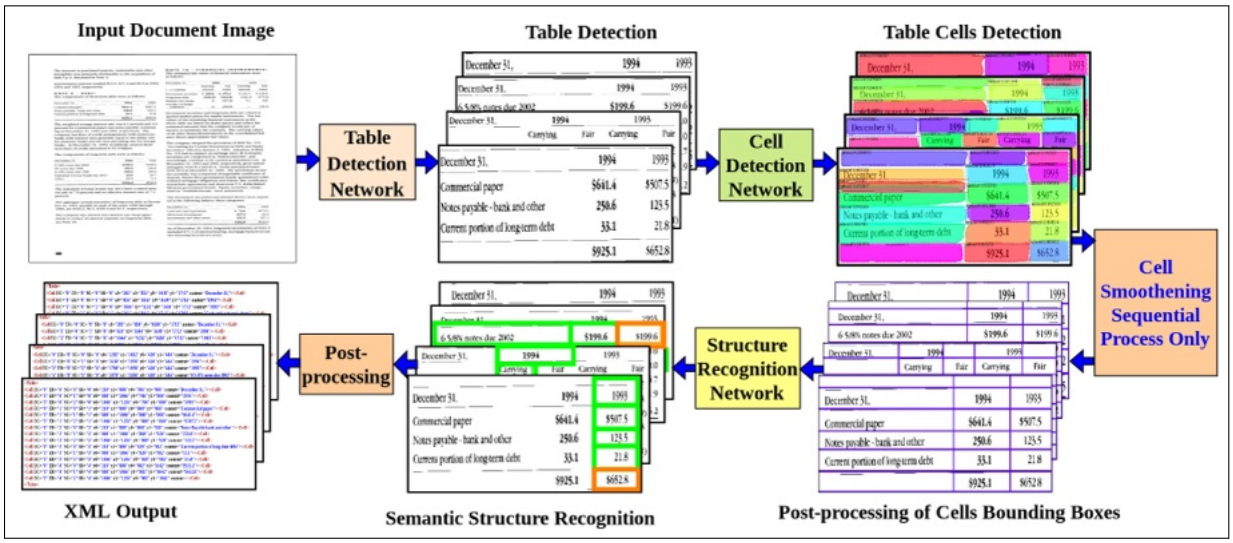


Figure 1.5: Block diagram of table understanding. Table detection is a precursor to table understanding. In the sequential process, table cells detection and structure recognition happen sequentially.

Despite its importance, table detection remains a challenging problem due to the vast diversity of document layouts and the variability in how tables are visually represented. Tables can appear with or without ruling lines, exhibit irregular or nested structures, and vary widely in cell spacing, alignment, and density. The presence of noise from scanning, skewed orientations, overlapping graphics, and variations in font or background texture further complicate the process. Moreover, in complex layouts such as scientific papers or financial reports, distinguishing tables from other visually similar elements like charts, lists, or text columns can be particularly difficult. The problem is further amplified in multilingual and historical documents, where degradation, handwriting, or non-standard printing conventions introduce additional ambiguity. Thus, table detection is not merely a segmentation task—it is a perceptual and contextual understanding challenge central to the automation of document intelligence.

Detecting tables with a high Intersection over Union (IoU) is critical because even small localization errors can have a cascading negative effect on downstream document understanding tasks. A table is not just a visual object—it encapsulates structured, relational data whose accurate extraction depends heavily on precise boundary detection. If a table is only partially detected (low IoU), important rows, columns, or headers may be excluded from subsequent processing, leading to incomplete or misleading information extraction. Conversely, if the detected region extends beyond the true table boundary, surrounding text or figures might be mistakenly included, introducing noise and confusion in structure recognition, cell segmentation, or content parsing stages. Therefore, high-IoU table detection ensures that the full spatial extent of the table is captured while minimizing contamination from non-table regions.

1.3.2 Structure Recognition through Cell Detection and Localization

Accurate cell detection forms the foundation for subsequent structure recognition [1], [2]. Unlike natural objects in scene images, table cells exhibit high visual similarity and depend heavily on context for disambiguation. A numeric value in one cell may be visually indistinguishable from a similar value in an adjacent cell, with only position and context indicating their distinct semantic roles [12]. Cell detection must achieve high localization accuracy, as even small misalignments can cause content to be incorrectly attributed to wrong cells during subsequent processing [2]. Standard intersection-over-union (IoU) thresholds of 0.5 commonly used in generic object detection are insufficient for table cells, where IoU values exceeding 0.8 or 0.9 may be necessary to ensure complete content capture [2]. This requirement for high-precision localization distinguishes table cell detection from most other object detection tasks [1].

Once cells are localized, structure recognition involves determining the row and column associations between all detected cells [1], [12]. This requires identifying which cells belong to the same row, which belong to the same column, and determining spanning relationships for cells that occupy multiple rows or columns [1], [2]. Structure recognition is inherently a relational reasoning task, as the identity of each cell is defined not by its intrinsic properties but by its relationships to other cells [12]. Graph-based formulations have proven effective for encoding these relationships, treating cells as nodes and adjacency relationships as edges [12], [24]. However, predicting adjacency relationships in dense tables with numerous cells results in quadratic complexity, necessitating efficient inference strategies [1]. Long-range dependencies pose additional challenges. In tables with many rows or columns, cells that are spatially distant may belong to the same logical grouping, requiring models to capture dependencies across the entire table structure [2]. Multi-row and multi-column spanning cells complicate adjacency prediction, as a single spanning cell may be adjacent to multiple cells in perpendicular directions [1].

1.3.3 Semantic Understanding of Tables

While Table Structure Recognition (TSR) provides the foundational physical and logical layout of a table, the ultimate goal of document intelligence is to achieve a deeper semantic understanding of its content [1], [31]. Semantic understanding moves beyond the mere identification of rows, columns, and cells to interpret the inherent meaning and relationships within the data. This includes discerning the hierarchical structure of headers, understanding the relationship between a data cell and its corresponding row and column headers, and recognizing implicit financial concepts such as totals, subtotals, and their constituent line items [2], [32].

One of the most effective paradigms for achieving and evaluating semantic table understanding is Visual Question Answering (VQA) [32], [33]. Beyond structural understanding, the ability to answer questions about tabular content represents a higher level of document comprehension [33]. Table VQA

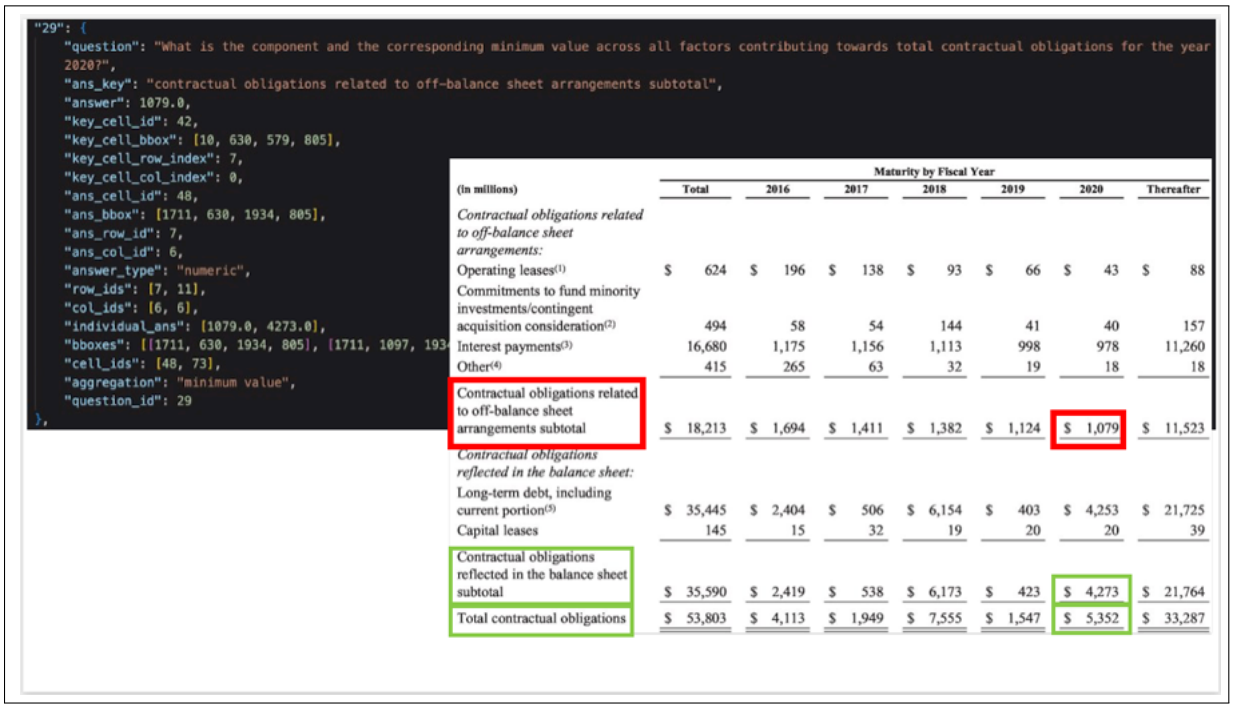


Figure 1.6: VQA on Table Images. Given a question and the cropped table image, output specific answer referring the contents of the table from the image.

encompasses multiple reasoning capabilities. Extractive questions require precise localization of specific cells based on row and column header understanding [33]. Numerical reasoning questions demand arithmetic operations such as computing ratios, percentages, or growth rates across multiple cells [34]. Aggregation questions involve operations like summation, averaging, or identifying extremal values across groups of cells [35]. Multi-hop reasoning questions require chaining multiple operations or synthesizing information from disparate table regions [32].

The challenges of table VQA are compounded by the visual nature of document images. Unlike structured databases where cell values are directly accessible, VQA on document images must contend with OCR errors, varying text rendering, and the need to precisely ground answers in visual regions [36]. Furthermore, business documents often contain domain-specific terminology, complex financial calculations, and hierarchical structures that require specialized knowledge for interpretation [33].

1.4 Why Solving Table Understanding Matters

The importance of robust table understanding extends far beyond academic interest, with substantial implications for practical applications across industries [3], [13].

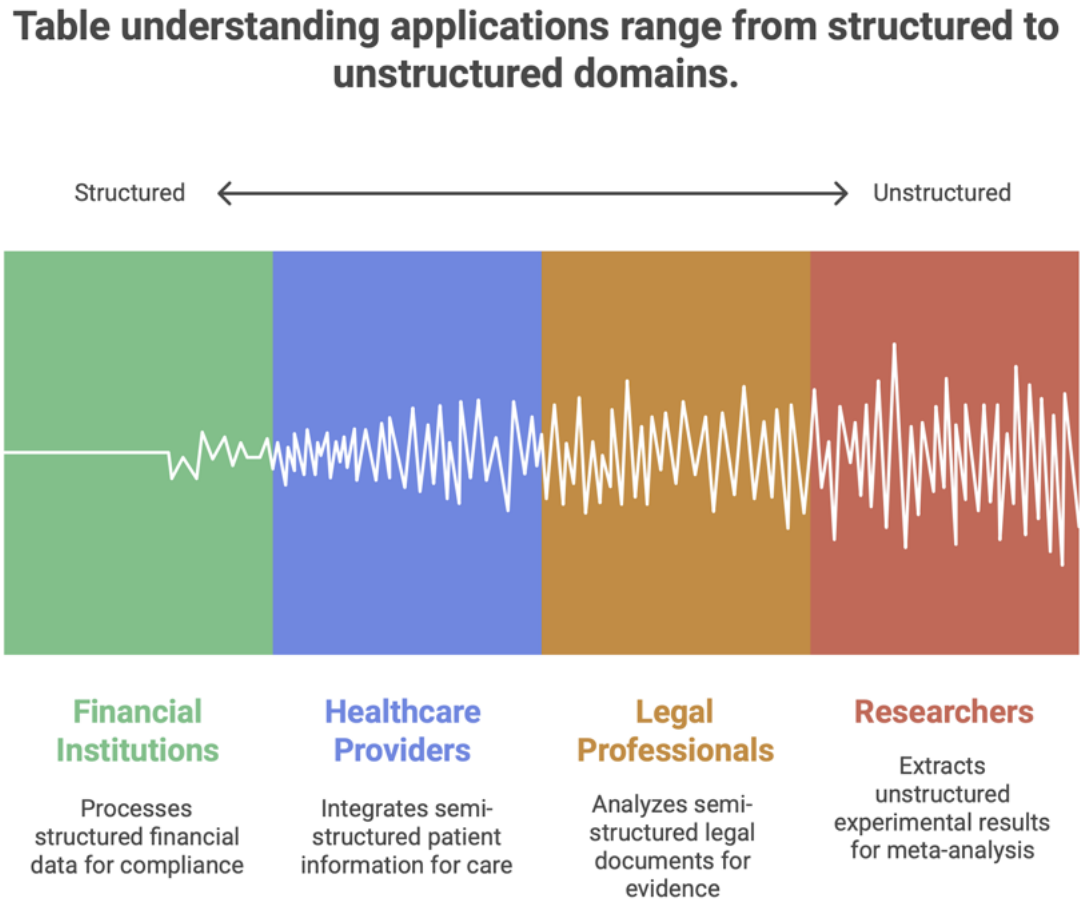


Figure 1.7: Table Understanding across domains

Financial Analysis and Regulatory Compliance Financial institutions process enormous volumes of tabular data from annual reports, earnings statements, regulatory filings, and internal financial records [37]. Credit underwriters must extract specific metrics from financial statements to assess creditworthiness and make lending decisions. Compliance officers need to verify regulatory reporting accuracy across thousands of documents. Portfolio managers require rapid extraction of financial indicators to inform investment strategies [37].

Manual processing of these documents is slow, expensive, and error-prone [38], [39]. Even minor transcription errors in financial data can lead to incorrect decisions with significant monetary consequences. Automated table understanding systems enable financial institutions to process documents at scale while maintaining accuracy and audit trails that satisfy regulatory requirements [37]. Evidence-based VQA systems provide the transparency necessary for regulatory compliance, allowing auditors to trace every computed metric back to source cells in original documents [32].

Scientific Literature Mining and Meta-Analysis Scientific publications across disciplines extensively use tables to present experimental results, demographic data, statistical analyses, and comparative evaluations [3]. Researchers conducting systematic reviews or meta-analyses must extract data from hundreds or thousands of published papers—a task that currently requires substantial manual effort [40].

Automated table understanding could accelerate scientific discovery by enabling large-scale extraction and aggregation of experimental results [40]. Question answering capabilities would allow researchers to query scientific literature interactively, asking questions like "What was the average effect size across studies using intervention X?" without manually reviewing each paper [40]. This capability would facilitate evidence synthesis, enable identification of research gaps, and support data-driven hypothesis generation across scientific domains [3].

Healthcare Data Management Electronic health record systems must integrate data from diverse sources including handwritten forms, typed reports, lab results, and historical paper records [3]. Medical tables contain critical patient information including vital signs, medication dosages, lab values, and treatment timelines. Accurate extraction and integration of this tabular data is essential for patient safety, continuity of care, and clinical decision support [3].

Question answering systems on medical tables could enable clinicians to quickly retrieve patient-specific information, track trends over time, and identify potential drug interactions or contraindications [36]. Evidence-based responses would provide the traceability required for medical documentation and legal accountability [32].

Legal Document Analysis Legal proceedings generate massive volumes of documentary evidence containing tables with financial records, timelines, organizational structures, and evidentiary exhibits [3]. Legal professionals must analyze these documents to identify relevant information, verify facts, and construct arguments. Automated table understanding can significantly reduce the time and cost associated with legal document review while improving thoroughness and accuracy [3].

VQA systems specialized for legal documents could answer specific questions about contracts, financial disclosures, or regulatory filings, dramatically accelerating discovery processes [33]. Evidence localization capabilities ensure that extracted information can be verified and cited in legal proceedings [32].

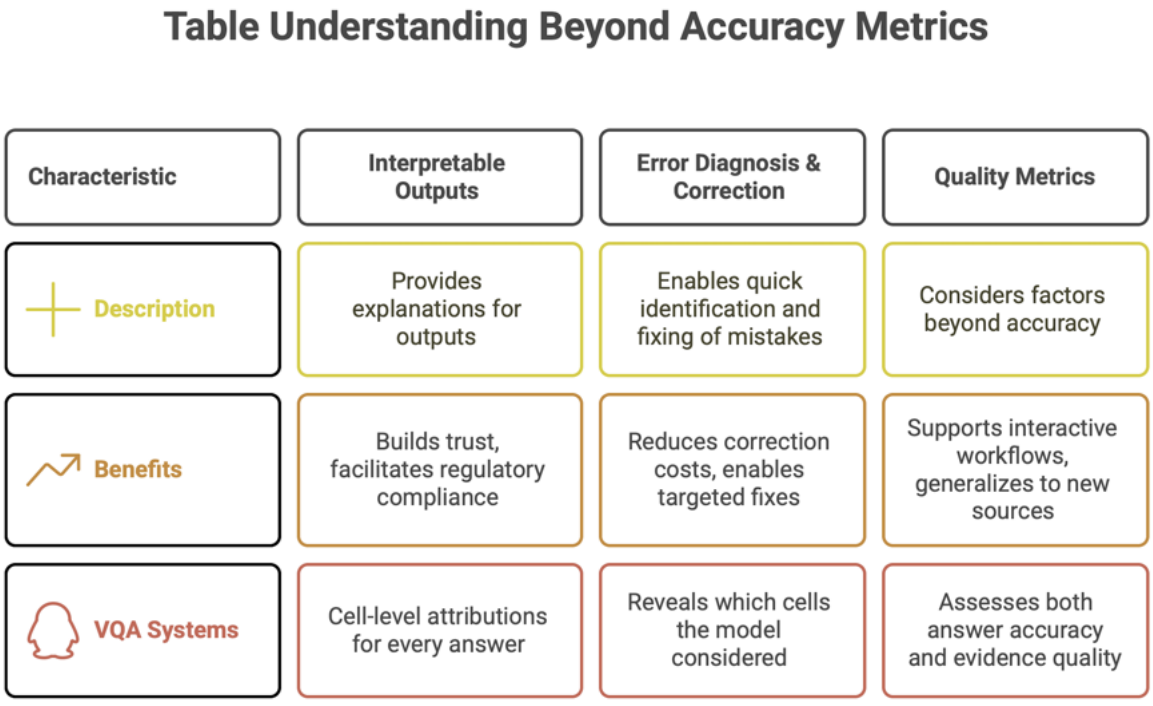


Figure 1.8: Requirements beyond accuracy in automated table understanding

1.5 Beyond High Accuracy

While achieving high accuracy in table reconstruction is essential, it is not sufficient for practical deployment in high-stakes applications [1], [2]. Real-world systems must support human verification, error diagnosis, and rapid correction when mistakes occur.

The Need for Interpretable Outputs Black-box table understanding systems that provide only final structured outputs without explainability limit their practical utility [2]. Domain experts reviewing automatically extracted tables need to quickly assess the reliability of the extraction and identify potential errors. Systems that provide confidence scores, attention visualizations, or intermediate outputs enable users to make informed judgments about whether to trust automated results or conduct manual verification [2].

For financial applications, regulatory frameworks often require audit trails demonstrating how automated systems arrived at particular outputs. Interpretable table understanding systems that can highlight which visual features influenced cell detection and structure prediction facilitate regulatory compliance and build user trust [37]. Evidence-based VQA systems inherently address this requirement by providing explicit cell-level attributions for every answer [32].

Error Diagnosis and Efficient Correction Even highly accurate systems will occasionally produce errors. The cost of correcting errors depends critically on how quickly mistakes can be identified and how easily they can be rectified [2]. Table understanding systems should provide structured outputs that enable efficient error correction interfaces.

For instance, if a system incorrectly merges two cells or fails to detect a spanning relationship, the correction interface should allow users to modify these specific structural attributes without requiring complete re-annotation of the entire table [1]. Hierarchical output representations that separate cell detection from structure prediction enable targeted corrections at the appropriate level of granularity [1], [2].

For VQA systems, error diagnosis requires understanding whether mistakes stem from OCR failures, incorrect cell localization, faulty reasoning logic, or misunderstanding of natural language queries [33]. Evidence-based systems facilitate this diagnosis by revealing which cells the model considered, allowing experts to quickly identify whether the error occurred in evidence selection or answer computation [32].

Quality Metrics Beyond Accuracy Traditional accuracy metrics such as cell detection F1-scores and structure recognition accuracy provide important but incomplete assessments of system utility [41], [42]. Practical deployments must consider additional dimensions of performance including processing speed, memory requirements, and robustness to input variations [1].

For VQA systems, evaluation must assess both answer accuracy and evidence quality [32]. A system that produces correct answers through incorrect reasoning paths is unreliable and potentially dangerous in high-stakes applications. Combined metrics that multiplicatively penalize failures in either answer correctness or evidence localization provide more meaningful assessments of real-world utility [32].

Processing speed determines whether systems can support interactive workflows or must operate in batch mode. Memory requirements influence deployment on edge devices versus cloud infrastructure. Robustness to variations in image quality, scanning resolution, and table layout conventions affects generalization to new document sources [2].

Comprehensive evaluation protocols should assess performance across these multiple dimensions, providing stakeholders with complete information to make informed deployment decisions [42], [43].

1.6 Motivation for This Research

Despite substantial progress in table understanding research, significant gaps remain between current capabilities and the requirements of practical applications [1], [2]. Existing methods often make sim-

plifying assumptions that limit their applicability to real-world scenarios. Many approaches assume the availability of OCR outputs or text bounding boxes as inputs [12], [24], [44]. While this simplifies the problem, it introduces dependencies on external systems whose errors propagate through the pipeline. OCR systems may misrecognize characters, fail to detect text in low-contrast regions, or incorrectly segment merged text [45]. Methods requiring OCR inputs inherit these limitations [12]. Other approaches target specific table formats or domains, achieving strong performance on particular datasets but failing to generalize across varied table layouts [18], [46]. Domain-specific methods designed for scientific publications may fail on financial documents with different formatting conventions. Systems optimized for tables with explicit ruling lines struggle with borderless tables common in business documents [1].

For VQA on tables, most existing benchmarks focus on web tables with clean digital structure or simplified synthetic tables [14], [16]. These datasets do not capture the complexity of real-world document images with irregular layouts, merged cells, OCR noise, and hierarchical financial structures [32], [33]. Furthermore, most VQA datasets evaluate only answer accuracy without assessing whether models attended to correct evidence regions [32]. The limited availability of large-scale training data with high-quality annotations constrains the development of data-hungry deep learning models [3], [13]. Manually annotating table cell boundaries and structure is extremely time-consuming and expensive. Creating evidence-annotated VQA datasets requires even greater effort, as each question-answer pair must be linked to precise bounding boxes of all relevant cells [32]. Automated annotation methods that extract structure from source PDFs often produce low-quality annotations that fail to capture alignment constraints and empty cells [2].

Furthermore, most existing benchmarks evaluate models primarily on average-case performance across entire test sets. This evaluation paradigm provides limited insight into model behavior on specific types of challenging tables such as those with high cell density, numerous spanning cells, or complex hierarchical structures [1], [2]. Understanding performance across table complexity dimensions is essential for targeted improvements.

Scope and Objectives *This thesis addresses fundamental challenges in table structure recognition and visual question answering from document images, with the overarching goal of developing robust, accurate, and practical systems for automated table understanding and evidence-based reasoning [1], [2], [32], [33].*

Our research encompasses three interconnected areas of table understanding. First, we focus on physical table structure recognition—the task of identifying all table cells and determining their spatial layout and structural relationships. We distinguish this from logical structure recognition, which additionally requires understanding semantic roles of cells (headers vs. data), and from complete semantic parsing which involves full content extraction and relationship modeling [1]. Second, we address visual

question answering on business document images, where systems must jointly understand natural language queries and complex tabular layouts to produce accurate answers [33]. Our VQA research spans multiple reasoning types including extractive queries, numerical computations, aggregation operations, and multi-hop reasoning over hierarchical financial structures [33]. Third, we investigate evidence-based visual question answering, where systems must not only generate correct answers but also provide explicit evidence through bounding box localization of all cells used in reasoning [32]. This capability is essential for audit-ready AI systems in financial analysis, regulatory compliance, and other high-stakes domains where transparency and verifiability are paramount [32].

The scope encompasses tables from diverse document types including scientific publications, financial reports, business documents, and historical archives. We address tables with varying degrees of complexity including borderless tables, tables with multi-row and multi-column spanning cells, dense tables with numerous cells, tables containing empty cells, and tables with hierarchical structures spanning multiple years or categories [1], [2], [32]. Our approach makes minimal assumptions about input documents. We require only document images as input for structure recognition, without relying on PDF metadata, OCR outputs, or domain-specific formatting conventions [1]. For VQA tasks, we accept natural language questions in addition to document images, enabling flexible interaction modalities [33]. This design choice maximizes generalizability across document sources and acquisition modalities while supporting practical deployment scenarios.

Throughout this research, we emphasize the dual requirements of accuracy and interpretability. Systems must not only perform well on benchmark metrics but also provide outputs that enable human verification, error diagnosis, and seamless integration into human-in-the-loop workflows [1], [32].

1.7 Key Contributions

This thesis makes several novel contributions to the fields of table structure recognition and visual question answering, spanning algorithmic innovations, architectural designs, training methodologies, evaluation protocols, and benchmark datasets.

Top-Down and Bottom-Up Framework We introduce a novel framework that combines top-down decomposition with bottom-up reconstruction for table structure recognition [1]. The top-down component employs object detection to localize individual table cells. The bottom-up component uses graph-based reasoning to predict adjacency relationships and reconstruct the complete table structure [1]. This decomposition provides several advantages. Cell detection as an object detection task benefits from extensive prior work in computer vision on accurate object localization. Graph-based structure recognition naturally represents the relational nature of table organization. The separation of concerns enables targeted optimization of each component and facilitates interpretability by providing intermediate out-

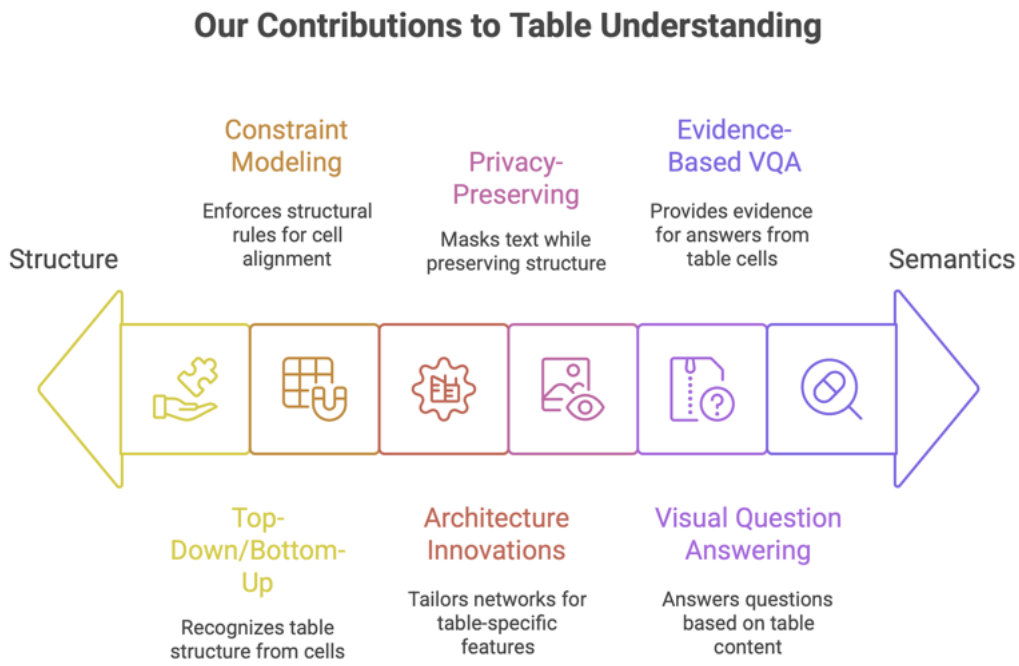


Figure 1.9: Contributions of this thesis work towards automated parsing and understanding of tables

puts [1], [2]. We demonstrate that this framework generalizes across diverse table layouts and document types, achieving state-of-the-art performance on multiple benchmark datasets [1], [2].

Constraint Modeling We formulate novel loss functions inspired by human visual perception of tables [1], [2]. Our alignment loss enforces that cells in the same row share aligned horizontal boundaries, while cells in the same column share aligned vertical boundaries. The continuity loss encourages that adjacent rows and columns exhibit spatial continuity without large gaps. The overlap loss penalizes intersecting cell boundaries, enforcing non-overlapping constraints [1], [2]. These auxiliary loss terms regularize the cell detection network to produce outputs that conform to structural properties of tables, even when individual cells lack distinctive visual features. Importantly, we introduce trainable loss weights that allow the model to learn appropriate relative importance of these constraints across different table regions [1], [2]. Extensive ablation studies demonstrate that structural constraint modeling significantly improves detection accuracy, particularly for challenging cases involving empty cells, dense layouts, and extreme aspect ratios [1], [2].

Architecture Innovations We propose several architectural innovations tailored to table structure recognition. Our cell detection network modifies standard object detection architectures to better handle the extreme aspect ratios and scale variations characteristic of table cells [1]. We incorporate attention

mechanisms that enable the model to focus on relevant structural features while suppressing irrelevant background information [2]. For structure recognition, we develop an efficient graph neural network that predicts row and column adjacencies between detected cells [1]. Our formulation uses rectilinear adjacency relationships rather than separate row and column adjacencies, simplifying post-processing and improving accuracy for spanning cells [2]. In our latest work, we introduce local attention-aware vision transformers as backbones for feature extraction from high-resolution table images. These hierarchical architectures generate multi-scale feature pyramids that effectively capture both fine-grained cell details and global table structure [4].

Privacy-Preserving Table Structure Recognition Recognizing that tables from sensitive domains such as healthcare, finance, and legal documents raise privacy concerns, we develop the first privacy-preserving framework for table structure recognition [47]. Our TabGuard system masks all textual content locally while preserving structural cues necessary for accurate cell detection and structure recognition. This enables secure cloud-based processing where the service provider never accesses sensitive data [47]. We propose an OCR-free text masking algorithm that operates efficiently on commodity hardware, enabling real-time content protection. We demonstrate that table structure can be accurately recognized from masked images, with performance comparable to methods operating on original unmasked images [47]. This contribution addresses critical privacy requirements for deploying table understanding systems in regulated industries.

Visual Question Answering on Business Documents Extending beyond structure recognition to higher-level semantic understanding, we organize and conduct the ICDAR 2023 Competition on Visual Question Answering on Business Document Images [33]. We create a large-scale VQA dataset on financial statements from FinTabNet, generating over 1.2 million question-answer pairs across five difficulty categories [33]. Our dataset generation methodology leverages automated heuristics combined with expert validation to create diverse questions including simple extraction queries, ratio calculations, temporal aggregations, hierarchical aggregations, and key-value extraction [33]. The competition attracted participants from leading research institutions and industry, establishing baseline performance levels and identifying key challenges in document VQA [33]. Analysis of submitted methods reveals that while modern vision-language models can handle extractive questions reasonably well, complex multi-step reasoning and numerical computations remain challenging [33]. This finding motivates continued research into specialized architectures and training procedures for document-centric VQA.

Evidence-Based Financial Visual Question Answering Recognizing the critical importance of interpretability and verifiability in financial applications, we introduce EviFiVQA, the first large-scale benchmark for evidence-based financial visual question answering [32]. This dataset contains over one million question-answer pairs on financial statement images, with each pair meticulously annotated with bounding boxes for all relevant table cells used to compute the answer [32]. EviFiVQA uniquely integrates the inherent hierarchical tree structures of financial documents to curate complex aggregation and

multi-hop reasoning questions [32]. The dataset encompasses real-world financial statements from S&P 500 companies spanning two decades, capturing diverse visual challenges including high-density tables, multi-span cells, nested structures, and complex multi-year layouts [32]. We develop a comprehensive evaluation protocol that jointly assesses answer accuracy and evidence quality [32]. Our evidence scoring mechanism uses IoU-based set matching to evaluate whether models correctly identify all relevant cells, while our answer scoring combines exact match accuracy with soft matching metrics to handle minor OCR variations [32]. The multiplicative combination of evidence and answer scores ensures that models must excel at both reasoning and spatial grounding to achieve high performance. Benchmark experiments with state-of-the-art vision-language models reveal significant limitations in current approaches [32]. While zero-shot models struggle severely with evidence localization, fine-tuned models show improvement but still fall far short of deployment-ready performance. Category-level analysis demonstrates that extractive questions are most tractable, while hierarchical aggregation and key-value extraction remain extremely challenging [32].

Training Data Curation and Augmentation Recognizing the challenges of acquiring high-quality annotated training data, we develop automated and semi-automated approaches for dataset creation. We propose algorithms that analyze source PDFs to extract cell boundaries while enforcing alignment and continuity constraints often violated by naive extraction methods [2]. We introduce a principled data curation strategy that balances training data across visual complexity dimensions including cell density, presence of spanning cells, and proportion of empty cells. This balanced training enables models to achieve consistent performance across the full spectrum of table complexity rather than overfitting to common cases [2]. For VQA datasets, we develop template-based question generation pipelines that leverage table structure annotations and domain knowledge to create diverse, realistic questions [32], [33]. We employ large language models for question paraphrasing to ensure linguistic diversity while preserving semantic meaning [32]. Automated consistency checks combined with expert validation ensure dataset quality at scale.

Comprehensive Evaluation Methodology We develop comprehensive evaluation methodologies that assess model performance across multiple dimensions of table complexity and reasoning difficulty [1], [2], [32]. Rather than reporting only aggregate metrics, we categorize test examples according to structural properties and analyze performance within each category. This fine-grained evaluation reveals strengths and weaknesses that guide future research directions. For structure recognition, we conduct extensive experiments across multiple benchmark datasets including ICDAR 2013 [48], ICDAR 2019 [41], UNLV [49], SciTSR [24], PubTabNet [50], TableBank [51], and FinTabNet [4]. This cross-dataset evaluation assesses generalization capabilities and identifies dataset-specific biases [1], [2].

For VQA evaluation, we employ difficulty-stratified assessment across question categories, enabling targeted analysis of model capabilities on different reasoning types [33]. We use weighted evaluation metrics that balance performance across categories to prevent systems from focusing exclusively on

high-frequency question types [33]. For evidence-based VQA, we introduce evaluation protocols that jointly measure answer correctness and evidence quality, ensuring that models must master both aspects to achieve high scores [32]. Our IoU-based evidence matching with set-level precision and recall provides nuanced assessment of evidence localization capabilities.

1.8 Outline of This Thesis

This thesis is organized to reflect a progressive research trajectory that evolves from foundational exploration of table structure recognition to advanced topics addressing privacy and explainability in document understanding. The overall structure of the thesis is designed to guide the reader from problem formulation and background concepts to a detailed exposition of proposed methodologies and their broader implications.[1]

Chapter 2 talks about the related work in the broader space of document AI, table detection, table structure recognition, table semantic understanding through representation learning and VQA, and privacy preservation in document AI. Chapter 3 introduces **TabStruct-Net**, a deep learning framework for Table Structure Recognition (TSR) that establishes the foundational methodology of decomposing table images into constituent cells followed by structural reconstruction through graph reasoning. Building upon this foundation, we present the intermediate architectures, **TOD-Net** and **TSR-Net**, which address the limitations of the earlier framework by incorporating multi-scale feature fusion and transformer-based reasoning to handle complex, visually diverse table layouts. Chapter 4 extends this line of work through **TabStruct-Net V2**, introducing a hierarchical local-attention vision transformer backbone that significantly improves scalability, robustness, and domain generalization.

The focus then shifts in Chapter 5 towards the imperative of **privacy preservation in Document AI**. Here, the proposed **TabGuard** framework pioneers privacy-preserving TSR through client-side anonymization and server-side structure inference, enabling secure deployment of TSR models on sensitive documents without exposing textual content. Chapter 6 broadens the scope from structure recognition to semantic reasoning through **EviFiVQA**, a benchmark for evidence-based financial visual question answering that establishes explainability and verifiable reasoning as core tenets for trustworthy AI in financial document understanding.

Finally, Chapter 7 consolidates the insights drawn from the preceding chapters, discussing the broader implications of this research in the context of explainable and privacy-preserving Document AI. These chapters provide a synthesis of contributions, highlight emerging research directions, and outline potential extensions towards scalable, auditable, and human-aligned document intelligence systems.

[1] The contents of the chapters are built on top of our publications [1], [2], [32], [33], [52], [53]

Chapter 2

Background and Related Work

2.1 Introduction to Tabular Data in Vision and Natural Language Processing

Tables are one of the most fundamental yet complex visual–linguistic structures used to convey structured information. From scientific papers and financial statements to web pages and invoices, tabular layouts provide a compact, human-readable medium that encodes multidimensional relationships between entities and attributes. Unlike free-form text or natural images, a table embodies both *spatial organization* and *semantic alignment*, requiring methods that integrate low-level vision, structural reasoning, and language understanding.

Significance of Tabular Data as a Modality. The prominence of tables in digital and scanned documents has motivated extensive research in their automated interpretation. Statistical reports, healthcare forms, and financial disclosures depend heavily on tables to present aggregated facts. Consequently, table understanding serves as a core enabler for document intelligence, knowledge retrieval, and visual question answering. According to [13], more than 30% of structured visual elements in business documents are tables. Yet, parsing a table remains far more challenging than detecting natural objects because it lacks consistent appearance cues and instead depends on spatial alignment, cell boundaries, and textual semantics.

Dual Nature: Vision and Language. Table understanding straddles two traditionally separate domains. In the **vision community**, research emphasises the detection of table regions in document images and the recovery of their geometric structure—rows, columns, and individual cells—using computer vision techniques [13], [46], [54]. Conversely, in the **natural-language community**, work has focused on reasoning over textual tables represented in HTML, CSV, or relational form, enabling question answering and semantic parsing [31], [55]. The convergence of these perspectives has given rise to *vision–language models for tables*, which aim to bridge visual perception and textual reasoning in a unified framework.

Challenges in Visual Table Understanding. Visually understanding tables in scanned or PDF documents is difficult for several reasons:

- **Layout variability:** Tables exhibit diverse grid structures—ruled, borderless, nested, or spanning—that defy simple heuristics.
- **Degraded quality:** Scans, noise, and skew make line detection unreliable.
- **Multi-scale structure:** A table may contain tiny cells or large merged regions that require multi-scale perception.
- **Semantic ambiguity:** Even if the grid is extracted, understanding header–data relations and units demands language reasoning.

Notation and Evaluation Measures. Evaluation of visual table understanding relies on several complementary metrics. Let B_p and B_g denote a *predicted* and a *ground-truth* bounding box, respectively. The *Intersection over Union* (IoU) is the most widely used measure of detection fidelity:

$$\mathrm{IoU}(B_p, B_g) = \frac{|B_p \cap B_g|}{|B_p \cup B_g|}, \tag{2.1}$$

A high IoU indicates that the detected region preserves the exact extent of the table, ensuring that subsequent structure recognition operates on complete visual evidence. In structure recognition, precision (P), recall (R), and F_1 score over cells or structural tokens are defined as

$$P = \frac{N_{\text{correct}}}{N_{\text{pred}}}, \qquad R = \frac{N_{\text{correct}}}{N_{\text{gt}}}, \qquad F_1 = \frac{2PR}{P+R}, \tag{2.2}$$

where N_{correct} counts correctly recovered items, N_{pred} is the total number of predicted items, and N_{gt} is the total number of ground-truth items. Precise structural reconstruction is critical because even small topological errors—for example, a missing split line or incorrect spanning cell—can distort downstream semantic reasoning and quantitative analysis.

Evolution of the Field. Early efforts in the 1990s and 2000s relied on handcrafted heuristics and ruled-line analysis [9], [19], [20], while the 2010s witnessed data-driven learning approaches that combined convolutional networks with document layout cues [5], [13], [46]. The recent shift toward transformer and graph-based models **liu2021neural**, [12], [56], [57] has enabled explicit modelling of cell relationships and large-scale pretraining on synthetic and annotated datasets. Most recently, large multimodal language models such as Qwen2-VL [58], DeepSeek-VL [59], and LLaMA-3 [60] have begun to transform document understanding by enabling zero-shot and instruction-tuned reasoning over table images without specialised training pipelines.

Scope of This Chapter. This chapter provides a comprehensive review of research on table detection, structure recognition, semantic understanding, and visual question answering over tables. We examine the evolution of approaches from early rule-based systems to recent transformer-based architectures. We explore semantic table understanding, including header linking and entity-relation inference. We also survey representation-learning paradigms that encode tables for reasoning. We then focus on Table VQA, where visual and textual reasoning converge. A dedicated subsection discusses modern OCR and document AI systems—including DeepSeek-OCR, Nanonets OCR, and Chandra OCR—that form the practical document ingestion backbone and contextualise the contributions of the thesis within the current state of the field. Finally, we discuss multimodal vision–language models, benchmarks, and outstanding challenges that define future directions and the importance of privacy in table understanding.

2.2 Table Detection and Table Structure Recognition

The task of *table structure recognition* (TSR) lies at the core of document image understanding. It involves two intertwined sub-problems: locating tables within complex page layouts (**table detection**) and recovering their internal grid structure of rows, columns, and cells (**structure recognition**). While early systems relied on handcrafted rules exploiting visual separators and text alignment, modern approaches employ deep neural architectures that model both geometry and semantics [13], [19], [20], [46], [54]. This section traces the chronological evolution of TSR, highlighting the progression from rule-based algorithms to CNN, graph, and transformer paradigms.

Early Heuristic and Rule-based Approaches (1990–2010). The earliest attempts at TSR arose within the document analysis community, preceding the widespread use of deep learning. Itonori [20] proposed one of the first algorithmic frameworks for recognising tables in printed documents by exploiting the spatial arrangement of text blocks and ruled lines. Kieninger [19] extended this notion through robust block segmentation and geometric heuristics, assuming that consistent inter-line spacing and alignment implied tabular structure. Hu et al. [8] introduced *medium-independent table detection*, emphasising that tables could appear across scanning or electronic sources, thus motivating layout-invariant representations. Later, Gatos et al. [54] and Kasar et al. [61] introduced rule-based systems leveraging horizontal and vertical line projection profiles to detect grid boundaries. These systems were sensitive to noise and could not handle borderless tables or irregular cell layouts but established the fundamental pipeline of: (1) line detection, (2) block grouping, and (3) cell segmentation.

A representative line-projection formulation computes the vertical projection profile $P_v(y)$ of an image $I(x, y)$ as

$$P_v(y) = \sum_{x=1}^{W} I(x, y), \tag{2.3}$$

where W denotes the image width and the summation integrates pixel intensities across each horizontal scan-line y. Peaks in P_v correspond to horizontal ruling lines, while valleys delineate row separations.

An analogous horizontal profile $P_h(x) = \sum_{y=1}^{H} I(x, y)$ captures vertical separators. Although simplistic, such formulations provided measurable cues for detecting grid lines in binarised document images.

OCR-driven and Hybrid Methods. Before the advent of deep learning, researchers combined optical character recognition (OCR) outputs with geometric reasoning. E Silva et al. [62] employed Hidden Markov Models (HMMs) to model table boundary sequences, treating successive textual tokens as observations conditioned on latent structural states (header, data, margin). Jahan et al. [63] reconstructed tables from OCR streams by detecting recurring alignment patterns in character baselines. Such hybrid systems improved robustness for low-contrast scans but were limited by OCR accuracy and the inability to represent non-rectilinear cell topologies.

Deep Learning for Table Detection (2015–2019). The resurgence of computer vision through convolutional networks revolutionised TSR. Gilani et al. [5] pioneered CNN-based detection of tables as generic visual objects, treating entire tables as bounding boxes within a page.

Input: A full document page image $I \in \mathbb{R}^{H \times W \times 3}$. **Output:** A set of axis-aligned bounding boxes $\{B_i\}$ with class labels $\{l_i\}$ indicating table regions.

Subsequent systems such as DeepDeSRT [46] integrated table detection and structure recognition into a unified network with two prediction heads: one for bounding boxes and another for cell segmentation masks. DeepDeSRT was trained on the ICDAR 2013 dataset [48], achieving significant improvement over classical methods.

Given predicted boxes $\{B_i\}_{i=1}^{n}$ and ground-truth boxes $\{G_j\}_{j=1}^{m}$, each prediction is evaluated via the IoU defined in Eq. 2.1. The mean Average Precision (mAP) across IoU thresholds quantifies detection performance:

$$\mathrm{mAP} = \frac{1}{T} \sum_{t=1}^{T} \mathrm{AP}_t, \qquad \mathrm{AP}_t = \int_0^1 P_t(r)\, dr, \tag{2.4}$$

where $P_t(r)$ denotes precision at recall r under IoU threshold $t \in [0, 1]$ and T is the number of IoU thresholds evaluated (commonly $T = 10$ for $t \in \{0.50, 0.55, \ldots, 0.95\}$). This formalism, borrowed from general object detection, became standard in table localisation.

Li et al. [13] introduced TableBank, the first large-scale benchmark containing 417k labelled table images mined from Word and LaTeX sources. Its scale enabled robust fine-tuning of generic detection architectures such as Faster R-CNN [30] and Mask R-CNN [29], which outperformed earlier single-page heuristics. TableSense [6] and other contemporaneous works focused on spreadsheet-like structures, proposing specialised convolutional backbones optimised for grid regularity.

Deep Splitting and Merging Networks. While CNN detectors localise tables, recovering individual cell boundaries requires fine-grained segmentation. Tensmeyer et al. [18] proposed a *Deep Splitting and Merging* architecture that performs row- and column-wise segmentation by predicting splitting points and subsequently merging adjacent regions using learned affinities.

Input: A table image $I_T \in \mathbb{R}^{H\times W\times 3}$. **Output:** Row split maps $S_r \in \mathbb{R}^H$ and column split maps $S_c \in \mathbb{R}^W$, together with a binary cell adjacency matrix $\hat{A} \in \{0,1\}^{N\times N}$, where N is the number of detected candidate cells.

The merging objective minimises the binary cross-entropy between predicted adjacency matrix $\hat{A}_{ij}$ and ground-truth A_{ij}:

$$\mathcal{L}_{\text{merge}} = -\frac{1}{N^2}\sum_{i,j}\Big[A_{ij}\log\hat{A}_{ij} + (1-A_{ij})\log(1-\hat{A}_{ij})\Big]. \tag{2.5}$$

This formulation laid the groundwork for graph-based reasoning, interpreting cells as nodes connected by adjacency edges.

Evaluation Metrics for Structure Recognition. Beyond detection, accurate structure reconstruction demands cell-level evaluation. The *Tree Edit Distance-based Similarity* (TEDS) metric introduced by Zhong et al. [3] compares the predicted table structure serialised as an HTML tree T_p with the reference tree T_g:

$$\text{TEDS}(T_p, T_g) = 1 - \frac{d_{\text{edit}}(T_p, T_g)}{\max(|T_p|, |T_g|)}, \tag{2.6}$$

where d_{edit} denotes the minimal number of node insertions, deletions, and substitutions required to transform T_p into T_g, and $|T|$ denotes the number of nodes in tree T. A TEDS score of 1 indicates perfect structural equivalence; a score of 0 indicates no structural overlap.

To further capture geometric alignment, Smock et al. proposed the *Grid Table Similarity* (GriTS) metric [64], which represents each table as a normalised grid of intersection points. Let $p_k^{(p)}$ be the k-th predicted grid intersection and $p_l^{(g)}$ be a ground-truth intersection. The similarity is given by

$$\text{GriTS} = 1 - \frac{1}{N}\sum_{k=1}^{N}\min_{l}\| p_k^{(p)} - p_l^{(g)} \|_2, \tag{2.7}$$

where N is the total number of predicted intersection points and $\|\cdot\|_2$ is the Euclidean distance. GriTS penalises even slight misalignments, making it sensitive to row/column shift errors.

Datasets and Benchmark Evolution. Public benchmarks have been instrumental in the progress of TSR. The ICDAR 2013 Table Competition [48] established the first standardised evaluation. Marmot and UNLV datasets introduced borderless tables and multi-page layouts. TableBank [13] scaled annotation volume dramatically, enabling large CNNs. PubTables-1M [65] later contributed one million high-resolution table images with cell-level annotations, bridging scanned and digital formats. FinTabNet and SciTSR extended the diversity toward financial and scientific domains, introducing extreme aspect ratios and spanning cells. Together, these datasets revealed persistent challenges: handling small fonts, merged headers, and domain-specific conventions such as currency or unit alignment.

Graph-based Reasoning for Table Structure Recognition (2019–2022). While convolutional networks successfully localised tables and segmented cells, they lacked explicit modelling of inter-cell dependencies. To address this, several works reconceptualised TSR as a *graph inference* problem, where nodes represent cells and edges encode adjacency relations.

Qasim et al. [12] introduced a graph neural network (GNN) framework that jointly models spatial and textual features.

Input: A set of N detected cell proposals, each represented by a node feature vector $\mathbf{h}_i^{(0)} \in \mathbb{R}^d$ derived from: (a) bounding box coordinates (x_i, y_i, w_i, h_i) linearly embedded, and (b) a visual embedding $\mathbf{v}_i$ extracted by a CNN crop of the cell image. Edges are initialised based on geometric proximity (IoU or distance thresholding).

Output: Updated node embeddings $\mathbf{h}_i^{(L)}$ after L message-passing layers, from which binary edge predictions $\hat{A}_{ij} = \sigma(\mathbf{h}_i^{(L)\top} W_e \mathbf{h}_j^{(L)})$ determine the cell adjacency graph.

Message passing updates node states through

$$\mathbf{h}_i^{(l+1)} = \sigma\!\left(\mathbf{W}_1 \mathbf{h}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}_2 \mathbf{h}_j^{(l)}\right), \tag{2.8}$$

where α_{ij} is the attention coefficient between nodes i and j (computed as a normalised dot product of their features), $\sigma(\cdot)$ is a non-linear activation (ReLU or GELU), $\mathcal{N}(i)$ is the set of neighbours of i, and $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ are learnable weight matrices. This enables the network to propagate context along structural boundaries, overcoming the locality limitations of CNNs.

Riba et al. [66] extended this paradigm for key-value invoice extraction, encoding spatial relations as directed edges and learning relational embeddings through graph convolutions. Liu et al. **liu2021neural** proposed the *Neural Collaborative Graph Machine* (NCGM), which integrates multi-head attention into graph propagation, effectively combining GNNs and transformers. With K attention heads indexed by k, the aggregated node embedding after L layers is

$$\mathbf{h}_i^{(L)} = \mathrm{Concat}_{k=1}^{K} \left(\sum_{j \in \mathcal{N}(i)} \mathrm{softmax}_j\!\big(\mathbf{q}_{i,k}^\top \mathbf{k}_{j,k}\big) \mathbf{v}_{j,k} \right), \tag{2.9}$$

where $\mathbf{q}_{i,k}, \mathbf{k}_{j,k}, \mathbf{v}_{j,k} \in \mathbb{R}^{d/K}$ are the query, key, and value vectors for the k-th attention head. Eq. 2.9 demonstrates how self-attention can generalise graph message passing by allowing each node to attend to all others, weighted by learned affinity.

Graph-based formulations explicitly represent adjacency and spanning relationships, permitting losses defined on edge existence:

$$\mathcal{L}_{\text{edge}} = -\sum_{(i,j)} \big[A_{ij} \log \hat{A}_{ij} + (1 - A_{ij}) \log(1 - \hat{A}_{ij}) \big], \tag{2.10}$$

analogous to Eq. 2.5, but extended to directed graphs. These methods achieved state-of-the-art accuracy on the SciTSR and PubTables-1M benchmarks, particularly for complex, borderless tables.

Transformer Architectures for TSR (2020–2024). Building on the success of attention mechanisms, transformer-based models have recently dominated TSR. Xue et al. [56] introduced TGRNet, the first end-to-end transformer for table grid reconstruction from images. The model encodes the visual features of each detected cell through a CNN backbone and applies multi-head self-attention layers to infer row and column relationships. The scaled dot-product attention follows the canonical form:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{2.11}$$

where $\mathbf{Q} \in \mathbb{R}^{N\times d_k}$, $\mathbf{K} \in \mathbb{R}^{N\times d_k}$, and $\mathbf{V} \in \mathbb{R}^{N\times d_v}$ are the query, key, and value matrices, d_k is the key dimensionality, and the softmax is applied row-wise. The factor $1/\sqrt{d_k}$ prevents dot-product magnitudes from growing excessively large. The contextualised embeddings enable long-range dependency modelling across widely separated cells.

Subsequent works such as LGPMA [67] and TableFormer [68] introduced layout-aware positional encodings. Each cell i is assigned a 2-D positional embedding $\mathbf{p}_i$ derived from its normalised coordinates $(x_i/W, y_i/H)$:

$$\mathbf{p}_i = [\sin(\omega x_i/W),\ \cos(\omega x_i/W),\ \sin(\omega y_i/H),\ \cos(\omega y_i/H)], \tag{2.12}$$

where ω is a frequency hyper-parameter. These embeddings are added to token features before self-attention, allowing spatial ordering to be preserved within the transformer encoder.

Anand et al. [57] proposed *TableCraft*, a unified multi-task model that jointly predicts table structure, cell types, and spanning configurations.

Input: A table image I_T rendered at high resolution. **Output:** (a) bounding boxes and class labels for each cell (detection head), (b) a binary cell-adjacency matrix $\hat{A}$ (edge head), and (c) a sequence of HTML structural tokens capturing spanning and hierarchical structure (text head).

It introduces a composite loss combining detection, adjacency, and textual alignment objectives:

$$\mathcal{L}_{\text{total}} = \lambda_1\mathcal{L}_{\text{det}} + \lambda_2\mathcal{L}_{\text{edge}} + \lambda_3\mathcal{L}_{\text{text}}, \tag{2.13}$$

where $\lambda_1, \lambda_2, \lambda_3 > 0$ are scalar weights balancing the contributions from structural detection ($\mathcal{L}_{\text{det}}$), graph-edge prediction ($\mathcal{L}_{\text{edge}}$, Eq. 2.10), and textual alignment ($\mathcal{L}_{\text{text}}$) objectives. By pretraining on large-scale synthetic tables and fine-tuning on PubTables-1M and FinTabNet, TableCraft achieved near-human TEDS scores and strong generalisation to unseen layouts.

Hybrid Architectures and OCR-free Variants. Traditional TSR pipelines depend on OCR text tokens to associate semantics with detected cells. Recent OCR-free models integrate visual and tex-

tual reasoning directly in the feature space. TableFormer-V2 [68] and TSRFormer [69] adopt hybrid encoder–decoder architectures where a vision transformer encoder produces patch embeddings and a sequence decoder predicts structural tokens in HTML-like syntax.

Input: A table image I_T. **Output:** A token sequence $T_{1:m}$ representing the HTML structure of the table (e.g., `<tr>`, `<td colspan="2">`, `</td>`, ...).

This paradigm treats TSR as a sequence-to-sequence problem:

$$P(T_{1:m} \mid I_T) = \prod_{t=1}^{m} P(T_t \mid T_{<t}, I_T; \theta), \tag{2.14}$$

where $T_{1:m}$ denotes the serialised table tokens, $T_{<t}$ is the prefix of already-generated tokens, and θ are the model parameters. Training minimises the negative log-likelihood with teacher forcing:

$$\mathcal{L}_{\text{seq}} = -\sum_{t=1}^{m} \log P(T_t^* \mid T_{<t}^*, I_T), \tag{2.15}$$

where T_t^* is the ground-truth token at position t. These OCR-free formulations demonstrate robust layout recovery even when text quality is poor, bridging the gap between document understanding and vision–language modelling.

Comparative Analysis of Model Paradigms. A comparison of representative TSR systems reveals a steady trend from local, heuristic reasoning toward global, contextual inference. Early rule-based methods [19], [20] focused on low-level geometry. CNN-based detectors (DeepDeSRT [46], TableBank [51]) learned discriminative visual patterns but ignored long-range dependencies. Graph-based networks (GNN-TSR [12], NCGM [70]) introduced explicit structural priors, while transformer architectures (TGRNet [56], TableCraft [57]) subsumed both vision and relation modelling in a unified attention mechanism. Empirically, transformer-based TSR models achieve TEDS ≥ 0.95 on PubTables-1M and GriTS ≥ 0.90 on SciTSR, surpassing all predecessors.

Remaining Limitations. Despite impressive progress, several challenges persist:

1. **Generalisation to unseen layouts:** Models often overfit to dataset-specific grid patterns and struggle with artistic or irregular tables.

2. **Scalability to high-resolution pages:** Vision transformers incur quadratic cost in image size; efficient sparse attention is still an open problem.

3. **Small-cell detection:** Cells with extreme aspect ratios or minute dimensions remain difficult to localise accurately, often leading to broken adjacency graphs.

4. **Lack of structural explainability:** Attention maps provide limited interpretability compared to explicit geometric reasoning.

Addressing these issues motivates research in semantic reasoning and multimodal fusion, topics discussed in the next sections.

2.3 Semantic Table Understanding

While table detection and structure recognition aim to recover the geometric skeleton of a table, *semantic table understanding* (STU) aspires to interpret the meaning of the extracted cells. It involves identifying the roles of headers, associating data with corresponding semantic concepts, and inferring relations among entities and attributes. The task bridges structural perception and natural-language reasoning, demanding multimodal representations that align textual content with spatial layout.

Problem Definition. Formally, let $\mathcal{T} = \{c_{ij}\}$ denote a table consisting of m rows and n columns, where each cell c_{ij} contains textual content t_{ij} and a bounding box $b_{ij} = (x_1, y_1, x_2, y_2)$. Semantic understanding seeks mappings

$$f : \mathcal{T} \to \{\text{Role, Entity, Relation}\}, \tag{2.16}$$

assigning every cell a functional role (e.g., header, stub, data), optionally linking it to an entity in an external knowledge base, and discovering binary or k-ary relations among cells.

Header Detection and Role Assignment. Early systems exploited heuristic alignment between header and data regions. Cafarella et al. [71] mined web tables by detecting header rows through capitalisation and string patterns.

In the deep-learning era, Nishida et al. [31] proposed a hybrid CNN–RNN network that learns visual cues for header zones jointly with textual semantics.

Input: A table $\mathcal{T}$ with cell images $\{I_{ij}\}$ and textual contents $\{t_{ij}\}$. **Output:** For each cell (i, j), a role label $r_{ij} \in \{\text{header, stub, data, caption}\}$.

Each cell embedding $\mathbf{h}_{ij}$ is computed as the concatenation of a visual feature vector $\mathbf{v}_{ij} \in \mathbb{R}^{d_v}$ extracted by the CNN and a contextualised word embedding $\mathbf{w}_{ij} \in \mathbb{R}^{d_w}$ obtained from an RNN over the cell text:

$$\mathbf{h}_{ij} = [\, \mathbf{v}_{ij};\, \mathbf{w}_{ij} \,] \in \mathbb{R}^{d_v + d_w}. \tag{2.17}$$

A bidirectional LSTM propagates context across rows and columns to predict role probabilities $p^r_{ij} = \mathrm{softmax}(\mathbf{W}_r \mathbf{h}_{ij})$. The training loss minimises cross-entropy between predicted and true role labels y^r_{ij}:

$$\mathcal{L}_{\text{role}} = -\sum_{i,j} \sum_{r} y^r_{ij} \log p^r_{ij}. \tag{2.18}$$

Entity Linking and Concept Normalisation. Beyond role detection, semantic interpretation often requires linking table cells to standardised concepts or database entities. Mulwad et al. [72] intro-

duced probabilistic models that align cell strings with DBpedia entities through lexical similarity and co-occurrence priors. More recent neural approaches (e.g., Deng et al. [55]) embed both cell text and candidate entity descriptions in a shared vector space. Let $\mathbf{e}_{ij} \in \mathbb{R}^d$ and $\mathbf{z}_k \in \mathbb{R}^d$ denote embeddings of cell (i, j) and candidate entity k, respectively. Entity assignment is formulated as a soft-matching problem:

$$P(e_{ij} = k) = \frac{\exp(\mathbf{e}_{ij}^\top \mathbf{z}_k / \tau)}{\sum_{k'} \exp(\mathbf{e}_{ij}^\top \mathbf{z}_{k'} / \tau)}, \tag{2.19}$$

where $\tau > 0$ is a temperature hyper-parameter that controls the sharpness of the distribution. The objective maximises the likelihood of correct alignments over training tables.

Relation Inference within Tables. Discovering relations among cells enables conversion of tables into knowledge-graph triples. Each relation can be represented as (h, r, t), where h and t are entity cells and r corresponds to their semantic relation (e.g., "Company $\rightarrow$ Revenue"). Neural models capture such relations by attending from data cells to headers. Bao et al. [73] introduced a table-to-text model that implicitly learns these alignments through attention weights:

$$\alpha_{ij} = \frac{\exp(\mathbf{q}_i^\top \mathbf{k}_j)}{\sum_{j'} \exp(\mathbf{q}_i^\top \mathbf{k}_{j'})}, \tag{2.20}$$

where $\mathbf{q}_i \in \mathbb{R}^d$ and $\mathbf{k}_j \in \mathbb{R}^d$ denote query and key vectors for data cell i and header cell j, respectively. The aggregated representation $\mathbf{r}_i = \sum_j \alpha_{ij} \mathbf{v}_j$ encodes the inferred relation context.

Table Classification and Type Prediction. Another dimension of semantic understanding is identifying the overall table type. Zhang et al. [74] categorised tables into relational, entity, and matrix types using convolutional embeddings of the HTML grid. Given an embedding $\mathbf{t} \in \mathbb{R}^d$ for the entire table, the classifier predicts $y = \mathrm{softmax}(\mathbf{W}\mathbf{t})$, trained under standard cross-entropy loss. Type prediction informs downstream reasoning: relational tables support SQL-like queries, whereas matrix tables encode 2D numerical relationships.

Hybrid Symbolic–Neural Reasoning. Hybrid approaches combine rule-based schema induction with neural representation learning. TURL [75] learns cell, column, and table embeddings via masked-language pretraining on relational tables. Each column embedding $\mathbf{c}_j$ is obtained through vertical self-attention:

$$\mathbf{c}_j = \mathrm{LayerNorm}\Big(\mathbf{h}_j + \mathrm{MHAttn}(\mathbf{h}_j, \mathbf{h}_j, \mathbf{h}_j)\Big), \tag{2.21}$$

where $\mathbf{h}_j$ aggregates cell embeddings in column j and $\mathrm{MHAttn}(\cdot)$ is standard multi-head attention (Eq. 2.11). This is followed by contextual aggregation across rows. The model supports tasks such as column type inference and entity linking with minimal supervision.

Evaluation Metrics for Semantic Understanding. Evaluating semantic alignment requires metrics beyond IoU or TEDS. Common measures include precision, recall, and F_1 at the cell-link level:

$$P_{\text{link}} = \frac{|\text{Correct Links}|}{|\text{Pred Links}|}, \qquad R_{\text{link}} = \frac{|\text{Correct Links}|}{|\text{Gold Links}|}, \qquad F_{1,\text{link}} = \frac{2P_{\text{link}}R_{\text{link}}}{P_{\text{link}} + R_{\text{link}}}. \tag{2.22}$$

For text-generation tasks such as table-to-text, metrics like BLEU [76], ROUGE [77], and CIDEr [78] are adopted. The BLEU score for an N-gram order is given by:

$$\text{BLEU} = \exp\left(\min\left(0,\ 1 - \frac{r}{c}\right) + \sum_{n=1}^{N} w_n \log p_n\right), \tag{2.23}$$

where r and c are the reference and candidate lengths in tokens, p_n denotes clipped n-gram precision, and $w_n = 1/N$ are uniform n-gram weights. Semantic plausibility is sometimes evaluated via human judgements of factual consistency or by automatically computing relation-matching accuracy.

Discussion and Limitations. Semantic table understanding remains an open challenge. Header–data misalignments, missing units, and implicit aggregation operations hinder automatic interpretation.

Moreover, existing datasets (e.g., WikiTableQuestions [79], TAT-QA [34]) primarily address textual tables, while visual table semantics in scanned documents remain under-explored. Cross-domain and multilingual semantic reasoning also lag behind structural extraction. These gaps motivate the development of unified models capable of leveraging vision, language, and structure jointly—bridging toward table-centric visual question answering, discussed next.

2.4 Table Representation Learning

Representation learning aims to encode the structural and semantic information of tables into vector spaces suitable for downstream tasks such as classification, retrieval, and question answering. Unlike ordinary textual embeddings, table representations must capture hierarchical dependencies among cells, columns, and headers while preserving the 2-D layout. This section surveys key paradigms for table representation learning across symbolic, neural, and multimodal contexts.

Symbolic and Statistical Representations. Prior to neural encoders, tables were represented through handcrafted statistical features. Typical descriptors included the number of rows and columns, ratio of numeric to textual cells, and positional heuristics of headers. Such features enabled lightweight classifiers for tasks like table type detection or layout similarity but failed to generalise beyond specific domains. Latent-semantic indexing (LSI) and topic modelling were sometimes applied to textual tables by treating each column as a document and cell contents as words, producing rudimentary semantic embeddings.

Neural Encoders for Textual Tables. The shift toward large-scale pretraining transformed table representation learning. TaBERT [80] and TAPAS [16] pioneered transformer models pretrained on millions of relational tables.

Input: A table $\mathcal{T}$ serialised as a flat token sequence, together with an associated natural-language query q. Each token x_i is embedded as $\mathbf{x}_i = \mathbf{E}_{\text{text}}(x_i)$. **Output:** Contextual embeddings $\mathbf{H} = \{\mathbf{h}_i\}$ that can be used for cell-selection or aggregation prediction.

The encoder computes

$$\mathbf{H} = \text{TransformerEncoder}(X + P), \tag{2.24}$$

where $X \in \mathbb{R}^{L\times d}$ is the token embedding matrix (over a sequence of L tokens), and $P \in \mathbb{R}^{L\times d}$ contains positional encodings derived from row and column indices. For TAPAS, cell and question embeddings interact through attention masking schemes that constrain the receptive field along rows or columns.

The pretraining objective combines masked-language modelling (MLM) and next-cell prediction. The MLM loss is

$$\mathcal{L}_{\text{MLM}} = -\sum_{i\in\mathcal{M}} \log P(x_i^* \mid x_{<i}, x_{>i}), \tag{2.25}$$

where $\mathcal{M}$ is the set of randomly masked token positions and x_i^* is the true token at position i. TaBERT additionally predicts column types and row alignments via contrastive losses, encouraging semantically similar columns to occupy proximate embedding regions.

Graph-based Table Embeddings. To preserve relational inductive bias, several methods represent tables as graphs [12], [56]. Each node corresponds to a cell or column, with edges capturing spatial adjacency or semantic relations. Given adjacency matrix $A \in \{0,1\}^{N\times N}$ and node feature matrix $X \in \mathbb{R}^{N\times d}$, a standard graph convolutional network (GCN) update is

$$\mathbf{H}^{(l+1)} = \sigma\Big(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}\mathbf{H}^{(l)}W^{(l)}\Big), \tag{2.26}$$

where $\tilde{A} = A + I_N$ is the adjacency matrix with added self-loops, $\tilde{D}$ is the corresponding degree matrix ($\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$), $W^{(l)} \in \mathbb{R}^{d\times d'}$ is a learnable projection, and σ is a non-linear activation. Graph attention networks (GAT) replace the uniform normalisation with learned coefficients α_{ij} (Eq. 2.8), yielding adaptive weighting of cell interactions. TURL [75] leverages such vertical and horizontal message passing to encode column semantics (Eq. 2.21).

Multimodal Visual–Textual Representations. For tables in scanned or image form, purely textual encoders are inadequate. Multimodal approaches combine visual backbones with language transformers. The LayoutLM family [81]–[83] introduced two-dimensional positional embeddings to encode text positions in documents.

Input: A document image I together with OCR-extracted tokens $\{x_i\}$ and their bounding boxes $\{b_i\} = \{(x_1^i, y_1^i, x_2^i, y_2^i)\}$. **Output:** Contextual embeddings $\{\mathbf{h}_i\}$ encoding both the identity and spatial position of each token.

Each token embedding is augmented by its bounding box coordinates:

$$\mathbf{x}_i^{\text{layout}} = \mathbf{x}_i^{\text{text}} + \text{Embed}(x_i^{\text{box}}), \tag{2.27}$$

enabling spatially aware attention. Subsequent versions (LayoutLMv3 [83], LiLT [84]) integrate image patches and token sequences in a single encoder, bridging the gap between vision and language.

In table contexts, models such as DiT [85] fine-tune these architectures on table-centric datasets. They employ multimodal pretraining losses combining masked language modelling and image–text alignment:

$$\mathcal{L}_{\text{ITA}} = -\log \frac{\exp(\text{sim}(\mathbf{v}, \mathbf{t})/\tau)}{\sum_{t'} \exp(\text{sim}(\mathbf{v}, \mathbf{t}')/\tau)}, \tag{2.28}$$

where $\text{sim}(\mathbf{v}, \mathbf{t}) = \mathbf{v}^\top \mathbf{t}/(\|\mathbf{v}\|\|\mathbf{t}\|)$ is the cosine similarity between a visual embedding $\mathbf{v} \in \mathbb{R}^d$ and a textual embedding $\mathbf{t} \in \mathbb{R}^d$, and $\tau > 0$ is the temperature. This contrastive formulation aligns visual cells with their textual descriptions, a cornerstone for OCR-free table understanding.

Aggregation Strategies. An ongoing challenge is aggregating cell-level embeddings into table-level representations without losing fine-grained structure. Common pooling operations include row, column, and global averages:

$$\mathbf{t}_{\text{row}} = \frac{1}{n} \sum_j \mathbf{h}_{ij}, \quad \mathbf{t}_{\text{col}} = \frac{1}{m} \sum_i \mathbf{h}_{ij}, \quad \mathbf{t}_{\text{table}} = \frac{1}{mn} \sum_{i,j} \mathbf{h}_{ij}. \tag{2.29}$$

Alternatively, attention-based pooling computes

$$\mathbf{t}_{\text{table}} = \sum_{i,j} \beta_{ij} \mathbf{h}_{ij}, \qquad \beta_{ij} = \frac{\exp(\mathbf{u}^\top \tanh(W\mathbf{h}_{ij}))}{\sum_{i',j'} \exp(\mathbf{u}^\top \tanh(W\mathbf{h}_{i'j'}))}, \tag{2.30}$$

where $\mathbf{u} \in \mathbb{R}^d$ and $W \in \mathbb{R}^{d \times d}$ are learnable parameters. Eq. 2.30 allows the model to emphasise informative regions such as headers.

Cross-modal Pretraining Objectives. Recent table encoders adopt multimodal pretraining schemes analogous to CLIP [86]. Given paired image and text representations $(\mathbf{v}_i, \mathbf{t}_i)$ for a batch of N samples, the InfoNCE loss is:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(\text{sim}(\mathbf{v}_i, \mathbf{t}_i)/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(\mathbf{v}_i, \mathbf{t}_j)/\tau)}. \tag{2.31}$$

This aligns visual and textual embeddings while contrasting against mismatched pairs, enabling table representations usable in both image and text modalities.

Comparative Analysis. Transformer-based encoders (TAPAS [16], TaBERT [80]) excel at reasoning over symbolic tables, while graph-based models (TURL [75], NCGM [70]) better capture structural dependencies. Multimodal encoders (LayoutLMv3 [83]) unify spatial and textual cues, crucial for scanned documents.

However, these models are resource-intensive and often rely on pretraining corpora dominated by English tables, limiting cross-lingual generalisation. Hybrid architectures that fuse structural graphs with vision–language pretraining represent the current frontier of research.

Summary. Table representation learning has evolved from handcrafted descriptors to sophisticated multimodal embeddings integrating geometry, text, and semantics. These representations underpin modern Table VQA and document-understanding pipelines, where the ability to jointly reason over visual and symbolic features determines end-task performance.

2.5 Table-based Visual Question Answering

Visual Question Answering (VQA) extends the traditional QA paradigm by requiring models to reason over visual as well as textual information. When the visual content is a table rather than a natural image, the problem—commonly termed *Table VQA*—requires a system to understand both the spatial organisation of cells and the semantics of their textual contents. Given a question q and a table image I_T, the objective is to predict an answer a:

$$a^* = \arg\max_{a \in \mathcal{A}} P(a \mid q, I_T; \theta), \tag{2.32}$$

where $\mathcal{A}$ is the candidate answer space and θ are model parameters. Unlike generic VQA over photographs, the reasoning chain in Table VQA typically combines textual lookup, numerical computation, and multi-cell aggregation, all grounded in a 2-D visual layout.

From Textual Table QA to Visual Table VQA. Early research on question answering over tables operated on structured textual formats such as HTML or CSV. The **WikiTableQuestions** dataset [79] pioneered this line by pairing natural-language questions with semi-structured tables from Wikipedia. Models such as Neural Enquirer [87] learned to map questions into executable programs operating on tables. Subsequent work including SQLNet [88] and RAT-SQL [89] improved semantic parsing accuracy by explicitly modelling relations among columns via relation-aware attention.

However, these textual Table QA systems assumed that cell values and headers were cleanly extracted. In contrast, scanned documents and PDFs require reasoning from pixels: identifying cells, reading their text, and associating them with the question. The emergence of *Table VQA* thus unified document layout analysis and reasoning. Representative tasks such as **TAT-QA** [34] and **FinQA** [35] introduced hybrid datasets combining textual and visual evidence from financial reports.

Problem Formulation. Formally, let I_T denote a table image and $\mathcal{O} = \{o_1, \ldots, o_N\}$ be OCR tokens with bounding boxes b_i. The model must compute an answer a that may involve a subset $\mathcal{S} \subseteq \mathcal{O}$ of supporting evidence. The probability of an answer is often decomposed as

$$P(a, \mathcal{S} \mid q, I_T) = P(a \mid \mathcal{S}, q, I_T)\ P(\mathcal{S} \mid q, I_T), \tag{2.33}$$

where the first factor corresponds to reasoning given the selected evidence $\mathcal{S}$, and the second to evidence selection. Equation 2.33 underlies most pipeline and end-to-end architectures.

Representative Datasets. Table VQA research accelerated with the release of several benchmarks:

- **InfographicVQA** [90]: Questions over infographic posters combining charts, icons, and tables; emphasises multi-modal reasoning.
- **ChartQA** [91]: Focuses on bar and line charts with tabular components, highlighting numerical reasoning.
- **DocVQA** [36]: Provides document-image questions, including tables, receipts, and forms.
- **TAT-QA** [34]: Targets financial reports where evidence spans multiple tables and textual paragraphs.
- **FinTabNet** [92]: A large-scale corpus for pretraining visual table models used later for Table VQA fine-tuning.

These datasets vary in annotation granularity—from single-cell answers to multi-hop computations. Typical answer types include *extractive* (copying a cell value) and *abstractive* (computing or synthesising a new phrase).

Architectural Taxonomy. Broadly, Table VQA models can be categorised into three families:

1. **Token-based models** operate directly on OCR text tokens augmented with spatial embeddings (e.g., LayoutLM-QA [81]).
2. **Layout-aware multimodal models** encode both image patches and token positions within transformers (e.g., TILT [93], DocFormer [94]).
3. **End-to-end OCR-free models** such as Donut [95] and Pix2Struct [96] predict answers directly from pixels, eliminating OCR errors.

Each paradigm balances trade-offs between textual precision and visual robustness. Token-based models excel at lexical understanding but inherit OCR noise; OCR-free approaches handle non-standard fonts yet may struggle with fine-grained numerical details.

Model Architectures for Table VQA. Modern Table VQA systems typically consist of three components: (1) an encoder that fuses visual and textual features, (2) a reasoning module that performs relational and arithmetic operations, and (3) an answer decoder or classifier that produces textual or numerical outputs.

Token Alignment and Multimodal Encoding. Token-based Table VQA models such as LayoutLMv2-QA [82] extend the LayoutLM [81] backbone by combining visual and spatial embeddings with textual tokens.

Input: OCR tokens $\{t_i\}_{i=1}^{L}$ (words extracted by OCR), image patches $\{v_k\}_{k=1}^{P}$ (from a CNN region-of-interest feature extractor), and corresponding bounding boxes $\{b_i\}$. **Output:** Contextual embeddings $\{\mathbf{h}_i\}$ over the joint token–vision sequence, used to answer classification, span-extraction, or generation queries.

The joint embedding for token i is computed as:

$$\mathbf{x}_i = \mathbf{E}_{\text{text}}(t_i) + \mathbf{E}_{\text{pos}}(b_i) + \mathbf{E}_{\text{vis}}(v_i), \tag{2.34}$$

where $\mathbf{E}_{\text{text}}$ maps token t_i to a d-dimensional embedding, $\mathbf{E}_{\text{pos}}$ encodes the four bounding-box coordinates $b_i = (x_1, y_1, x_2, y_2)$ into a spatial embedding of the same dimension, and $\mathbf{E}_{\text{vis}}$ extracts a visual feature from the image region corresponding to v_i. The concatenated sequence is processed by a transformer encoder, which models dependencies between question words, visual regions, and cell tokens via cross-attention.

Layout-aware Attention Mechanisms. DocFormer [94] introduced 2-D relative position bias terms Δx_{ij} and Δy_{ij} for each token pair (i, j):

$$\text{Attention}_{ij} = \frac{(\mathbf{q}_i^\top \mathbf{k}_j) + \mathbf{b}_x(\Delta x_{ij}) + \mathbf{b}_y(\Delta y_{ij})}{\sqrt{d_k}}, \tag{2.35}$$

where $\Delta x_{ij} = x_i^{\text{centre}} - x_j^{\text{centre}}$, $\Delta y_{ij} = y_i^{\text{centre}} - y_j^{\text{centre}}$, and $\mathbf{b}_x(\cdot), \mathbf{b}_y(\cdot)$ are learnable scalar bias functions of relative displacement. This bias term enforces that spatially adjacent cells attend more strongly, preserving table geometry within attention layers.

Numerical and Symbolic Reasoning. Many Table VQA questions require arithmetic computation rather than direct extraction. TAPAS [16] introduced differentiable aggregation operators by extending the transformer output with scalar prediction heads.

Input: Contextual token embeddings $\mathbf{H} \in \mathbb{R}^{L \times d}$. **Output:** An aggregation operation type in $\{\text{SUM}, \text{AVG}, \text{COUNT}, \text{NONE}\}$ and a soft cell-selection mask $\{s_i\} \in [0, 1]^L$.

Given the [CLS] token embedding, aggregation logits $\mathbf{z} = W_{\text{agg}}[\text{CLS}]$ parameterise soft selection over operations:

$$P(\text{op}) = \text{softmax}(\mathbf{z}), \qquad a = \text{op}\big(\text{SelectedCells}\big). \tag{2.36}$$

During training, supervision for aggregation type is derived from weak labels or execution results. FinQA [35] and MathQA-style models [97] further extend this with symbolic equation generation, decoding explicit arithmetic expressions.

Program Generation and Execution. Program-based models map questions to symbolic programs that operate on table cells. Let $y_{1:T}$ denote a sequence of program tokens (e.g., `SELECT`, `ADD`, `ARG1`); the model maximises:

$$\mathcal{L}_{\text{prog}} = -\sum_{t=1}^{T} \log P(y_t^* \mid y_{<t}^*, q, I_T). \tag{2.37}$$

An executor then interprets the program over the table to produce the final answer. Neural-symbolic hybrids achieve high interpretability and robustness on compositional questions.

Multi-hop and Cross-table Reasoning. Datasets such as TAT-QA [34] and MultiHiertt [98] require reasoning across multiple tables and text segments. To handle this, models employ hierarchical attention or graph propagation across tables. Let each table T_k have representation $\mathbf{t}_k$ (Eq. 2.30); cross-table reasoning computes:

$$\tilde{\mathbf{t}}_k = \mathbf{t}_k + \sum_{l \neq k} \alpha_{kl} W \mathbf{t}_l, \qquad \alpha_{kl} = \frac{\exp(\mathbf{t}_k^\top \mathbf{t}_l)}{\sum_{l'} \exp(\mathbf{t}_k^\top \mathbf{t}_{l'})}, \tag{2.38}$$

where $W \in \mathbb{R}^{d \times d}$ is a learnable projection and α_{kl} is the attention weight from table k to table l. This mechanism integrates complementary evidence across multiple tables or surrounding text, enabling multi-hop question answering.

Answer Decoding Strategies. Depending on the dataset, answers may be categorical, extractive, or generative.

Classification-based decoding. For multiple-choice or yes/no questions, a classification head predicts the label:

$$P(a) = \text{softmax}(W[\text{CLS}]). \tag{2.39}$$

Span extraction. For extractive answers, start and end indices of the relevant text span are predicted:

$$P_{\text{start}}(i) = \text{softmax}(W_s \mathbf{h}_i), \quad P_{\text{end}}(i) = \text{softmax}(W_e \mathbf{h}_i), \tag{2.40}$$

where $W_s, W_e \in \mathbb{R}^{1 \times d}$ are learnable weight vectors.

Generative decoding. OCR-free and multimodal models like Donut [95] and Pix2Struct [96] treat answering as text generation conditioned on visual features (Eq. 2.14), using autoregressive decoders.

Loss Functions. A composite loss is typically employed combining answer, reasoning, and localisation terms:

$$\mathcal{L}_{\text{total}} = \lambda_a \mathcal{L}_{\text{ans}} + \lambda_r \mathcal{L}_{\text{reason}} + \lambda_e \mathcal{L}_{\text{evidence}}, \tag{2.41}$$

where $\lambda_a, \lambda_r, \lambda_e > 0$ are scalar weights balancing answer accuracy, reasoning correctness, and evidence localisation objectives, respectively. Evidence supervision is often derived from human-annotated bounding boxes or weakly aligned OCR tokens.

Intermediate Reasoning Supervision. To improve interpretability, some methods incorporate intermediate supervision in the form of rationales or cell-highlight masks. Given annotated evidence cells $\mathcal{S}^*$, a binary mask $\hat{m}_i$ is predicted for each token with cross-entropy loss:

$$\mathcal{L}_{\text{mask}} = -\sum_i [m_i^* \log \hat{m}_i + (1 - m_i^*) \log(1 - \hat{m}_i)]. \tag{2.42}$$

Multimodal and Vision–Language Models for Table VQA. Recent progress in large vision–language models (VLMs) has reshaped Table VQA by enabling end-to-end reasoning directly from pixels and text. Models such as PaLI-X [99], LLaVA [100], and Phi-3 Vision [101] unify visual and textual modalities through joint pretraining objectives that extend beyond natural images to include documents and tables.

Encoder–Decoder Framework.

Input: A table image I_T and a natural-language question q (tokenised). **Output:** A variable-length answer string $y_{1:T}$ generated autoregressively.

A vision encoder $\mathcal{E}_v$ extracts patch embeddings and a language decoder $\mathcal{D}_t$ generates the answer sequence. The generation objective is

$$\mathcal{L}_{\text{gen}} = -\sum_{t=1}^{T} \log P(y_t^* \mid y_{<t}^*, q, \mathcal{E}_v(I_T); \theta), \tag{2.43}$$

which generalises Eq. 2.15.

Vision Encoders. Document-specific VLMs often replace generic ViTs with layout-aware encoders such as DiT [102] or LayoutLMv3 [83]. A typical patch embedding pipeline is

$$\mathbf{v}_i = W_p \,\text{Flatten}\big(I_T[x_i : x_i + p,\ y_i : y_i + p]\big) + \mathbf{p}_i, \tag{2.44}$$

where p is the patch size in pixels, $I_T[\cdot]$ denotes a $p \times p$ image crop, $W_p \in \mathbb{R}^{d \times 3p^2}$ is a linear projection, and $\mathbf{p}_i \in \mathbb{R}^d$ is a 2-D positional encoding (Eq. 2.12).

Cross-modal Fusion. Cross-attention layers connect the vision and text streams:

$$\text{Attn}_{vt} = \text{softmax}\Big(\frac{\mathbf{Q}_t \mathbf{K}_v^\top}{\sqrt{d_k}}\Big)\mathbf{V}_v, \tag{2.45}$$

where $\mathbf{Q}_t = \mathbf{T}W_Q$ originates from textual token features $\mathbf{T}$, and $\mathbf{K}_v = \mathbf{V}_{\text{vis}}W_K$, $\mathbf{V}_v = \mathbf{V}_{\text{vis}}W_V$ from visual embeddings $\mathbf{V}_{\text{vis}}$. This mechanism allows question tokens to attend selectively to relevant table regions.

OCR-free Instruction-tuned Models. Models such as Pix2Struct [96] and UReader [103] eliminate OCR dependencies entirely by pretraining on rendered documents. Instruction-tuning further adapts these models for question answering, captioning, and reasoning tasks:

$$\mathcal{L}_{\text{inst}} = -\sum_{(I,q,a)} \log P(a \mid q, I; \theta_{\text{VLM}}). \tag{2.46}$$

Evaluation Metrics for Table VQA. *Exact-match.* The simplest criterion is exact string match (EM):

$$\text{EM} = \begin{cases} 1, & \text{if } a_{\text{pred}} = a_{\text{gold}}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.47}$$

Minor lexical variations motivate relaxed string and numerical matching, computed by normalised edit distance or value equivalence.

Execution accuracy. For program-based systems:

$$\text{Acc}_{\text{exec}} = \frac{1}{N}\sum_i \mathbf{1}\big[\text{Exec}(p_i) = a_i^*\big]. \tag{2.48}$$

Evidence F_1 and grounding. To measure faithfulness of evidence selection:

$$F_{1,\text{evidence}} = \frac{2|\mathcal{S}_p \cap \mathcal{S}_g|}{|\mathcal{S}_p| + |\mathcal{S}_g|}, \tag{2.49}$$

where $\mathcal{S}_p$ and $\mathcal{S}_g$ are the sets of predicted and ground-truth evidence cells.

Reasoning-step accuracy. Recent benchmarks (e.g., MultiHiertt [98], EviFiVQA [32]) propose multi-step evaluation:

$$\text{Acc}_{\text{step}} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}[s_k^{\text{pred}} = s_k^{\text{gold}}], \tag{2.50}$$

where K is the number of reasoning steps and s_k is the output of step k.

Pretraining and Fine-tuning Strategies. *Pretraining.* Common objectives include masked-language modelling (Eq. 2.25), image–text contrastive loss (Eq. 2.28), and visual-textual reconstruction losses. Pretraining on diverse document domains—scientific, financial, and business—improves layout generalisation.

Fine-tuning. During fine-tuning, question–answer supervision is introduced through the total loss (Eq. 2.41). Curriculum strategies gradually increase reasoning complexity, beginning with extractive questions and progressing to multi-hop numerical tasks.

Ablation Findings and Comparative Analysis. Empirical studies reveal several consistent patterns:

- Spatial-textual fusion and attention bias yield significant gains in localisation accuracy.
- Explicit reasoning supervision improves interpretability without degrading final accuracy.
- OCR-free models excel on high-resolution tables but underperform on small fonts compared to token-based hybrids.
- Large-scale instruction tuning, as in PaLI-X and LLaVA, enhances generalisation to unseen domains, suggesting the benefit of shared multimodal representations.

Summary. Table VQA unifies advances in table detection, semantic parsing, and vision–language modelling. It represents one of the most challenging forms of multimodal reasoning—requiring perception, structure induction, and symbolic computation within a single framework. The next section discusses general multimodal pretraining strategies for tables and documents, which provide the foundation for Table VQA systems.

2.6 Multimodal and Vision–Language Models for Tables

Motivation. While early table understanding pipelines relied on independent vision and language components, recent advances in multimodal pretraining have shown that a unified encoder trained on large-scale image–text pairs can learn cross-modal correspondences transferable to structured documents. For tables, this entails aligning spatial layouts, visual features, and textual semantics in a single representational space.

Foundations of Multimodal Pretraining. Let I denote a document or table image and T the corresponding textual sequence (OCR tokens or captions). Multimodal pretraining optimises an objective that encourages *modality alignment* and *joint understanding*. A typical composite loss combines three components:

$$\mathcal{L}_{\text{multi}} = \lambda_1 \mathcal{L}_{\text{MLM}} + \lambda_2 \mathcal{L}_{\text{ITM}} + \lambda_3 \mathcal{L}_{\text{CL}}, \tag{2.51}$$

where $\mathcal{L}_{\text{MLM}}$ is masked-language modelling (Eq. 2.25), $\mathcal{L}_{\text{ITM}}$ is image–text matching, and $\mathcal{L}_{\text{CL}}$ is a contrastive loss (Eq. 2.28).

Image–text matching. Given encoded image $\mathbf{v}$ and text $\mathbf{t}$ features, the model predicts whether they originate from the same document:

$$\mathcal{L}_{\text{ITM}} = -\sum_{(I,T)} [y \log \sigma(\mathbf{v}^\top \mathbf{t}) + (1-y) \log(1 - \sigma(\mathbf{v}^\top \mathbf{t}))], \tag{2.52}$$

where $y \in \{0, 1\}$ indicates whether (I, T) is a matched pair and σ is the sigmoid function.

Layout-aware Vision–Language Models. LayoutLMv2–v3 [81], [83] extend BERT [104] with visual embeddings obtained from CNN or ViT backbones. Each token is augmented with bounding-box coordinates and visual region features $\mathbf{v}_i$:

$$\mathbf{h}_i^{(0)} = \mathbf{E}_{\text{text}}(t_i) + \mathbf{E}_{\text{pos}}(x_1, y_1, x_2, y_2) + \mathbf{E}_{\text{vis}}(\mathbf{v}_i). \tag{2.53}$$

Through joint attention, the encoder learns to associate spatially co-located text tokens across rows and columns. Fine-tuning LayoutLM on Table VQA [35], [79] and DocVQA [36] improves performance on both structure recognition and question answering tasks.

Vision Transformer-based Document Models. DocFormer [94] introduced a hierarchical transformer that processes both patch embeddings and text tokens with 2-D positional biases (Eq. 2.35). TILT [93] employs a similar encoder–decoder structure optimised for document generation tasks such as question answering and summarisation. LiLT [84] achieves cross-lingual robustness by decoupling textual embeddings from language-specific vocabularies, enabling multilingual table processing.

Large Vision–Language Models (VLMs). Recent foundation-scale models (PaLI-X [99], LLaVA [100], Phi-3 Vision [101]) generalise multimodal learning to trillions of image–text pairs. They utilise a contrastive–captioning dual objective that simultaneously minimises $\mathcal{L}_{\text{CL}}$ (Eq. 2.28) and a generation loss (Eq. 2.43). For table-centric fine-tuning, visual encoders are exposed to document-rendered tables, while language decoders are instruction-tuned on mixed question–answer corpora covering textual, tabular, and chart data.

A simplified contrastive–captioning objective is:

$$\mathcal{L}_{\text{VLM}} = \mathcal{L}_{\text{CL}} + \beta\, \mathcal{L}_{\text{gen}}, \tag{2.54}$$

where $\beta > 0$ balances embedding alignment and generative fidelity.

LLM-based Systems for Table Structure Recognition and VQA. The emergence of instruction-tuned and chain-of-thought-capable LLMs has opened new pathways for table understanding that go beyond task-specific fine-tuning. This subsection discusses the most relevant recent systems.

Qwen2-VL [58]. Qwen2-VL is a large-scale vision–language model featuring a *Naive Dynamic Resolution* mechanism that processes table images at varying resolutions without fixed patch-size assumptions.

Input: A table image I_T (at arbitrary resolution) paired with a free-form question q. **Output:** A generated answer string that may reference specific cells, perform aggregation, or describe structural layout.

A key architectural innovation is the use of *2D-RoPE* (Rotary Position Embedding) to encode absolute spatial coordinates of each visual token, enabling the model to reason about row/column positions

directly. The model scales from 2B to 72B parameters and achieves state-of-the-art performance on DocVQA and structured document benchmarks. For table understanding, Qwen2-VL demonstrates strong OCR-free cell reading and cross-table numerical reasoning, making it a practical competitor to specialised TSR pipelines.

DeepSeek-VL [59]. DeepSeek-VL couples a high-resolution visual encoder with the DeepSeek LLM backbone, designed to balance *visual and language understanding through a careful data mixing strategy*.

Input: Document images (up to 1024×1024 pixels) and instruction prompts. **Output:** Free-form text answers, including structured information extraction and table reading.

The architecture employs a hybrid visual encoder that encodes low-resolution global context and high-resolution local patches separately, then fuses them before the language decoder. On document-oriented benchmarks such as DocVQA, DeepSeek-VL achieves competitive performance with substantially fewer parameters than GPT-4V, demonstrating the value of architecture-level optimisation for document understanding.

LLaMA-3 [60]. Although LLaMA-3 is primarily a text LLM, its instruction-tuned variants and derivative multimodal extensions (e.g., LLaMA-3-Vision) have been applied to table-reasoning tasks via structured prompting. Given a serialised table (HTML or markdown) as input text, LLaMA-3 can perform multi-step numerical reasoning, entity extraction, and schema inference with high accuracy on textual table benchmarks such as WikiTableQuestions and TAT-QA. Its strength lies in the combination of scale (8B–70B parameters), RLHF alignment, and strong zero-shot generalisation, which collectively allow the model to handle diverse table layouts expressed in text form.

Pixtral 12B [105]. Pixtral introduces a natively multimodal architecture with a *flexible-resolution ViT* that can process arbitrarily sized document images without resizing-induced distortion—important for high-aspect-ratio tables.

Input: A document image of arbitrary size and a text instruction. **Output:** A free-form text response, including structured content extraction.

The model uses *break tokens* to separate row patches within its visual token sequence, enabling the language decoder to maintain row-column correspondence. Pixtral achieves strong results on MMMU and DocVQA, and its flexible resolution design is particularly well-suited to tables that span multiple columns or contain dense numerical grids.

Complementary Role in the Thesis. The systems described above—Qwen2-VL, DeepSeek-VL, LLaMA-3, Pixtral—represent powerful general-purpose foundations that excel when sufficient compute is available and when prompting is feasible. The methods proposed in this thesis are complementary in several key ways. First, TabStruct-Net [1] and the proposed neurosymbolic TSR system target *explicit structural output* (adjacency graphs, cell bounding boxes) rather than free-form text, enabling deterministic post-processing in downstream pipelines. Second, TabGuard is designed for *privacy-preserving inference*: it operates on encrypted or anonymised document representations, a regime in which large VLMs cannot function without sharing raw data with cloud endpoints. Third, EviFiVQA introduces

an *evidence-grounded evaluation* that requires models to justify their answers with bounding-box-level annotations—a level of accountability that black-box VLMs do not natively provide. Thus, the thesis contributions occupy a complementary niche: structural correctness, privacy compliance, and interpretable reasoning that general-purpose LLM systems do not address.

Cross-modal Attention and Fusion. Cross-modal transformers unify table image and text representations by interleaving modality-specific attention blocks. Given vision features $\mathbf{V} \in \mathbb{R}^{N_v \times d}$ and text features $\mathbf{T} \in \mathbb{R}^{N_t \times d}$, a fusion layer computes:

$$\mathbf{F} = \mathrm{softmax}\left(\frac{(\mathbf{T}W_Q)(\mathbf{V}W_K)^\top}{\sqrt{d_k}}\right)\mathbf{V}W_V, \tag{2.55}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$ are learnable projections. The result $\mathbf{F} \in \mathbb{R}^{N_t \times d}$ contains text-conditioned visual features, followed by feed-forward projection to a shared space. Such layers allow questions about visual tables to selectively focus on relevant spatial regions, a prerequisite for Table VQA performance.

Multimodal Pretraining Corpora. Large-scale corpora underpin these models:

- **CCPDF** [106] and **DocLayNet** [107]—synthetic and real PDF renderings for layout learning.
- **TableBank** [51], **PubTables-1M** [65], and **FinTabNet** [3]—table images with cell-level annotations for supervised alignment.
- **InfographicVQA** [90] and **ChartQA** [91]—extend the domain to non-rectangular table formats.

Pretraining on such corpora exposes models to diverse fonts, languages, and structures, enhancing cross-domain generalisation.

Comparative Analysis. Compared to specialised TSR or Table VQA models, multimodal VLMs exhibit superior zero-shot adaptability but require massive compute and memory resources. LayoutLM-style encoders remain efficient for fine-tuning on moderate datasets, while PaLI-X or Phi-3 Vision offer broader transfer but less interpretability. The trade-off between generality and faithfulness continues to shape research on document-centric foundation models.

Summary. Multimodal and vision–language models have transformed table understanding by fusing textual, spatial, and visual signals within a unified transformer backbone. These models serve as the backbone for end-to-end Table VQA systems, where pretraining on document-level objectives yields powerful representations that generalise across domains.

2.7 Modern OCR and Document AI Systems

Optical Character Recognition (OCR) forms the foundational layer of any document AI pipeline that operates on scanned or photographed documents. Accurate OCR determines the quality of textual tokens available to downstream table-understanding modules; errors in recognition propagate to cell content, header text, and numerical values, directly affecting structure recognition and VQA performance. This section surveys the state of modern OCR infrastructure—from open-source engines to commercial cloud platforms—and discusses their relevance to the thesis contributions.

Classical and Neural OCR Foundations. The Tesseract OCR engine [45], originally developed at HP and later maintained by Google, pioneered open-source document OCR based on connected-component analysis and an adaptive classifier. Tesseract v5 introduced neural network line recognition using LSTM-based models, substantially improving accuracy on degraded documents. The docTR toolkit [108] provides an end-to-end deep-learning OCR pipeline combining EAST-style text detection with CRNN-based recognition, packaged for integration into Python document-AI workflows.

DeepSeek-OCR. DeepSeek-OCR is a high-accuracy OCR system developed by DeepSeek AI as part of their broader document understanding research programme. It is built on transformer-based text detection and recognition modules fine-tuned on large-scale multilingual document corpora.

Architecture. DeepSeek-OCR follows a two-stage pipeline. A detection backbone (based on a Feature Pyramid Network over a ResNet encoder) identifies text regions as oriented bounding polygons. A recognition transformer then processes each cropped text region as a sequence prediction task, mapping the image patch $I_{\text{crop}} \in \mathbb{R}^{H' \times W' \times 3}$ to a character sequence via an autoregressive decoder:

$$P(c_{1:L} \mid I_{\text{crop}}) = \prod_{l=1}^{L} P(c_l \mid c_{<l},\ \mathcal{E}_{\text{vis}}(I_{\text{crop}})), \tag{2.56}$$

where c_l is the l-th recognised character and $\mathcal{E}_{\text{vis}}$ is the visual encoder.

Relevance to the Thesis. DeepSeek-OCR's high accuracy on dense financial and scientific documents makes it directly relevant as a pre-processing front-end for the table structure recognition and VQA systems described in this thesis. In the TabStruct-Net and EviFiVQA pipelines, OCR-extracted tokens serve as the textual input modality; substituting a stronger OCR engine reduces token-level noise and consequently improves cell-content alignment. Furthermore, the evidence-grounded annotation framework in EviFiVQA depends on reliably extracted bounding boxes for each textual token—a quality that modern neural OCR systems like DeepSeek-OCR significantly improve upon classical rule-based engines.

Nanonets OCR. Nanonets OCR is a commercial deep-learning-based document processing platform designed for real-world enterprise deployment. It specialises in *structured document extraction*, with particular strength in invoice, receipt, and financial-table parsing.

Architecture. Nanonets OCR employs a multi-task neural architecture that jointly performs text detection, character recognition, and layout analysis within a unified inference graph. Its visual backbone processes documents at variable input resolutions and outputs bounding-box annotations at the word level, together with confidence scores. For table-specific inputs, the system includes a specialised *table extraction module* that identifies row and column separators and associates cell content with their grid coordinates—effectively performing a form of implicit table structure recognition.

Relevance to the Thesis. Nanonets represents a practitioner baseline against which the privacy-preserving recognition system (TabGuard) must be benchmarked. Nanonets processes raw document images on cloud servers, which raises data-sovereignty and privacy concerns when applied to confidential financial or medical documents. In contrast, TabGuard is explicitly designed to operate on encrypted or locally processed representations. The comparison between Nanonets-style systems and the TabGuard approach thus directly motivates the privacy-preservation contribution of the thesis: high accuracy should not require the sacrifice of data confidentiality.

Chandra OCR. Chandra OCR is a specialised Indian-language OCR system developed for processing documents containing Devanagari and other Indic scripts. It addresses a significant gap in the document AI landscape: most commercial OCR engines are optimised for Latin-script documents, while government forms, financial disclosures, and health records in India frequently use Hindi, Bengali, Tamil, Telugu, and other scripts.

Architecture. Chandra OCR employs a script-agnostic feature extractor based on a dilated convolutional encoder that captures fine-grained stroke patterns, followed by a language model adapted to Devanagari syllabic structure. The model handles *compound characters* (matras and conjuncts) through a structured prediction layer that models character-level dependencies beyond simple CTC decoding.

Relevance to the Thesis. The ICDAR 2023 competition dataset [33] and the EviFiVQA benchmark [32] include financial documents sourced from Indian corporate filings, which may contain mixed-script content. Integrating a system such as Chandra OCR as a front-end for Indic document sections would extend the thesis contributions toward a multilingual document AI pipeline—a direction identified in Section 2.10 as a critical future research axis. More broadly, Chandra OCR illustrates that the "solved" problem of OCR remains far from complete when the scope extends beyond English-Latin documents, reinforcing the need for language-inclusive evaluation in TSR and Table VQA research.

Positioning Thesis Contributions Relative to Modern OCR. A central question for any document AI thesis is: how do the proposed methods interact with, depend on, or improve upon the OCR layer? Three positions are possible—*OCR-dependent*, *OCR-free*, and *OCR-agnostic*—and the thesis contributions span all three.

- **TabStruct-Net (OCR-dependent):** The neurosymbolic table structure recognition system uses OCR tokens as a textual modality alongside visual features. Its accuracy is therefore bounded by the quality of the OCR engine. The evaluation in the thesis uses docTR [108] and Tesseract [45] as baselines; replacing either with DeepSeek-OCR or Nanonets OCR would be expected to reduce cell-content errors and improve the TEDS scores reported.
- **TabGuard (OCR-agnostic via encryption):** Privacy-preserving table recognition operates on encrypted visual features, bypassing any plaintext OCR extraction. The system is therefore compatible with any OCR engine at inference time, as long as textual features are encrypted before being fed to the model.
- **EviFiVQA (OCR-dependent for annotation, OCR-free for some baselines):** The benchmark construction relied on high-quality OCR to associate question evidence with bounding boxes. The evaluation includes both OCR-dependent (LayoutLMv3) and OCR-free (Donut, Pix2Struct) baselines. Modern OCR systems such as DeepSeek-OCR could serve as the extraction backbone for future extended versions of the dataset, improving annotation quality for low-contrast financial typographies.

In summary, the thesis does not propose a new OCR system; instead, it treats OCR as an existing infrastructure layer and focuses on the structural and reasoning capabilities that sit above it. However, the quality of that infrastructure—whether it is a classical engine like Tesseract, a commercial platform like Nanonets, a research system like DeepSeek-OCR, or a specialised tool like Chandra OCR—directly determines the practical ceiling of system performance and must therefore be acknowledged in the evaluation discussion.

2.8 Evaluation Benchmarks and Metrics

Introduction. The evolution of table understanding research has been shaped largely by the availability of benchmarks and standardised evaluation metrics. Datasets determine not only the representational biases of models but also the kind of reasoning they are able to perform. This section surveys the major benchmarks spanning Table Structure Recognition (TSR), Semantic Table Understanding (STU), and Table-based Visual Question Answering (Table VQA), followed by an overview of evaluation metrics used for localisation, structure recovery, and semantic reasoning.

Benchmarks for Table Detection and Structure Recognition. Early datasets such as the **ICDAR 2013 Table Competition** [48] and **Marmot** [109] introduced scanned scientific articles containing ruled and borderless tables. Annotations were provided in terms of bounding boxes and cell adjacency graphs. Subsequent datasets scaled both volume and diversity:

- **TableBank** [13]: over 417k table images from Word and LaTeX sources with bounding boxes for detection tasks.

- **SciTSR** [110]: scientific articles featuring complex spanning cells and irregular structures.
- **PubTables-1M** [65]: one million high-resolution tables annotated with structure trees and cell boundaries.
- **FinTabNet** [92]: financial statements with merged cells and numeric headers emphasising high-aspect-ratio challenges.

Benchmarks for Semantic Table Understanding.

- **WikiTableQuestions** [79]: natural-language questions and executable logical forms over HTML tables.
- **Spider** [111]: complex SQL query generation from text, covering cross-domain relational schemas.
- **TURL** [75]: a large corpus of relational tables with cell-to-entity mappings for pretraining column and table embeddings.
- **SQA** [112] and **WTQ-sup** [14] variants focus on conversational and compositional QA.

Benchmarks for Table VQA and Multimodal Reasoning.

- **TAT-QA** [34]: financial-report images requiring cross-table and paragraph reasoning.
- **InfographicVQA** [90]: infographic posters combining charts and tabular data.
- **ChartQA** [91]: synthetic and real charts with numeric questions.
- **DocVQA** [36]: general document VQA with table components.
- **EviFiVQA** [32]: a fine-grained benchmark for evidence-grounded financial Table VQA, annotated with answer bounding boxes and reasoning chains.

Evaluation Metrics for Structure Recognition. *Intersection over Union (IoU).* For table detection and cell segmentation:

$$\mathrm{IoU}(B_p, B_g) = \frac{|B_p \cap B_g|}{|B_p \cup B_g|}. \tag{2.57}$$

Detection quality is summarised via mAP (Eq. 2.4).

Tree Edit Distance Similarity (TEDS). For structural fidelity, TEDS (Eq. 2.6) computes normalised tree-edit similarity between predicted and ground-truth HTML trees.

Grid Table Similarity (GriTS). GriTS (Eq. 2.7) measures spatial alignment between grid intersection points, providing fine-grained geometric evaluation.

Metrics for Semantic Understanding. Semantic alignment and relational inference are assessed using link-level F_1:

$$F_{1,\text{link}} = \frac{2\,|\text{Correct Links}|}{|\text{Pred Links}| + |\text{Gold Links}|}. \tag{2.58}$$

For text-to-SQL or logical-form generation tasks, *Execution Accuracy* evaluates correctness of the executed query outcome:

$$\text{Acc}_{\text{exec}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\big[\text{Exec}(q_i^{\text{pred}}) = \text{Exec}(q_i^{\text{gold}})\big]. \tag{2.59}$$

Metrics for Table VQA. Evaluation in Table VQA incorporates answer correctness, reasoning trace, and evidence alignment:

- *Exact Match (EM) / Relaxed EM*: answer string compared after normalisation.
- *Execution Accuracy*: used when models output symbolic programs (Eq. 2.37).
- *Evidence* F_1: measures alignment between predicted and annotated evidence cells.
- *GriTS-Evidence* F_1: combines spatial and semantic matching for bounding-box evidence.
- *Step-Accuracy*: evaluates intermediate reasoning steps in multi-hop settings.

Benchmark Analysis and Limitations.

1. *Domain imbalance.* Most benchmarks focus on English-language scientific or financial tables, limiting cross-domain generalisation.
2. *Weak supervision.* Many datasets lack cell-level or evidence annotations, impeding fine-grained interpretability.
3. *Metric inadequacy.* Current metrics emphasise correctness over reasoning transparency; small format differences can inflate structural penalties despite semantic equivalence.
4. *Lack of unified protocol.* Evaluation pipelines vary across works, hindering reproducibility and fair comparison.

Emerging Directions.

- *Holistic Document Understanding Benchmarks* (e.g., DocLayBench) integrate table, chart, and text understanding within unified protocols.
- *Faithfulness-oriented Metrics* such as *Step Fidelity* and *Evidence Consistency Score* evaluate alignment between reasoning traces and answers.
- *Synthetic Data Augmentation* via table renderers expands layout diversity for more robust model assessment.

Summary. Evaluation benchmarks form the empirical backbone of table understanding research. From geometric IoU to reasoning-step accuracy, each metric captures a facet of model competence. True progress requires multidimensional evaluation that balances structural correctness, semantic validity, and explainability.

2.9 Privacy Preservation in Document AI

Document AI refers to the suite of techniques for extracting information from documents, including tasks such as layout analysis, form understanding, and table structure recognition. These tasks often involve processing sensitive documents like financial forms, medical records, or identity documents, where privacy is imperative. Unlike generic computer vision, Document AI systems frequently handle personally identifiable information (PII) and confidential data (e.g., patient health information or financial details). Ensuring privacy in Document AI is therefore crucial, both to comply with regulations (such as GDPR and HIPAA) and to maintain user trust. This section provides a literature review of privacy-preserving techniques applicable to Document AI.

We give a general overview of privacy concerns in Document AI, then delve into major paradigms: homomorphic encryption, federated learning, differential privacy, secure multi-party computation, and lightweight on-device methods (including cryptographic techniques such as elliptic curve cryptography). Each subsection reviews key academic works and discusses how the technique applies to document analysis, with mathematical formulations that clarify the approach.

Homomorphic Encryption. Homomorphic Encryption (HE) enables computations on encrypted data. For encryption function $\mathsf{Enc}(\cdot)$ and decryption $\mathsf{Dec}(\cdot)$, HE schemes ensure:

$$\mathsf{Dec}\big(\mathsf{Enc}(x_1) \otimes \mathsf{Enc}(x_2)\big) = x_1 \oplus x_2, \tag{2.60}$$

where $\otimes$ denotes a ciphertext-domain operation and $\oplus$ the corresponding plaintext operation. CNN-based HE systems like HCNN perform secure inference on encrypted data [113]. The feasibility of secure object detection was first demonstrated in cloud-based applications [114], and extended to deep models [115], [116].

Federated Learning. Federated Learning (FL) allows collaborative model training without centralising data. Each client i holds local dataset D_i and local loss $L_i(w)$. The global loss is:

$$L(w) = \frac{1}{N} \sum_{i=1}^{N} L_i(w). \tag{2.61}$$

Federated averaging (FedAvg) computes the global update as a weighted average of local model parameters:

$$w^{(t+1)} = \sum_{i=1}^{N} \frac{|D_i|}{\sum_j |D_j|} w_i^{(t)}, \tag{2.62}$$

where $w_i^{(t)}$ denotes the parameters of client i's model after local training step t. FL is effective for object detection [117]–[119] and applicable to layout and form analysis in document data.

Differential Privacy. Differential Privacy (DP) provides provable privacy guarantees. A mechanism $\mathcal{M}$ is (ε, δ)-DP if, for any neighbouring datasets D, D' differing in one record and any measurable output set S:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta, \tag{2.63}$$

where $\varepsilon > 0$ is the *privacy budget* (smaller is more private) and $\delta \geq 0$ is a small failure probability. In DP-SGD, each per-sample gradient is clipped and noised:

$$g_B = \frac{1}{|B|} \sum_{x_i \in B} \text{clip}(\nabla_w L(x_i), C) + \mathcal{N}(0, \sigma^2 C^2 I), \tag{2.64}$$

where C is the gradient clipping threshold, σ is the noise multiplier, and $|B|$ is the mini-batch size. Approaches include private neural activations [120], mixed private–public training [121], and scalable DP-SGD [122]. FL can be combined with DP for added privacy [123].

Secure Multi-Party Computation. Secure Multi-Party Computation (SMPC) enables joint computation without revealing inputs. Techniques include secret sharing and garbled circuits. Recent works applied SMPC to secure object detection [124]–[126]. Document AI can use MPC in settings with distributed data holders collaborating on tasks like layout analysis or fraud detection.

On-Device Inference and Lightweight Cryptography. MobileNet-based models [127]–[129] enable efficient inference on edge devices, allowing privacy-preserving layout and form detection locally. Elliptic Curve Cryptography (ECC) ensures secure communications and efficient key exchange.

Discussion. Each method balances privacy and utility differently. HE and SMPC offer strong cryptographic guarantees, while FL and DP focus on secure collaborative training. On-device methods favour practical privacy by keeping data local. Hybrid approaches combine these techniques for robust privacy-preserving Document AI systems.

2.10 Challenges, Gaps, and Future Directions

Overview. Despite significant progress, table understanding remains one of the most complex problems in document intelligence. Unlike natural images, tables encode structured symbolic relations that

require geometric reasoning, textual comprehension, and numerical logic simultaneously. This section outlines the persisting challenges that limit current approaches and highlights potential research directions toward more generalisable and explainable Table VQA systems.

Challenge 1: Layout Diversity and Domain Generalisation. A fundamental obstacle in table understanding is the extreme variability of layouts across documents and domains. Tables may be ruled, borderless, multi-level, nested, or even handwritten. Deep models trained on datasets such as TableBank or PubTables-1M often overfit to stylistic regularities of those corpora. Domain shift—such as moving from academic papers to financial statements—causes severe performance degradation.

Future work should explore **domain-adaptive pretraining** and **style-invariant representations**. Meta-learning and contrastive domain alignment can mitigate overfitting by optimising for cross-layout consistency. Given embeddings $\mathbf{h}_i^d$ from domains $d \in \{A, B\}$, domain invariance can be regularised by minimising:

$$\mathcal{L}_{\text{domain}} = \sum_i \|\mathbf{h}_i^A - \mathbf{h}_i^B\|_2^2, \tag{2.65}$$

thus encouraging feature uniformity across visual domains.

Challenge 2: Low-resource and Multilingual Table Understanding. Most existing datasets are English-centric. However, global applications—such as government reports or invoices—require multilingual OCR and cross-lingual semantics. Multilingual document understanding is hindered by language-specific typography and numerical formats (e.g., the Indian numbering system, Arabic scripts).

Possible directions include:

- **Multilingual pretraining:** Extend LayoutLM and TAPAS with multilingual corpora (mTAPAS, mLiLT).
- **Cross-lingual alignment losses:**

$$\mathcal{L}_{\text{align}} = -\sum_{(x^{L_1}, x^{L_2})} \log \frac{\exp(\text{sim}(\mathbf{h}_{x^{L_1}}, \mathbf{h}_{x^{L_2}})/\tau)}{\sum_{x'} \exp(\text{sim}(\mathbf{h}_{x^{L_1}}, \mathbf{h}_{x'})/\tau)}, \tag{2.66}$$

  where x^{L_1} and x^{L_2} are semantically equivalent tokens in languages L_1 and L_2.

- **Font-agnostic visual pretraining:** leveraging large-scale synthetic renderings of multilingual tables to overcome script scarcity.

Challenge 3: Numerical and Logical Reasoning. Current models struggle with arithmetic precision and compositional reasoning. For example, correctly answering "What is the average profit growth between 2018 and 2020?" may require multi-step computation involving row selection, numeric parsing, and division.

Hybrid neuro-symbolic architectures remain a promising avenue. By combining neural perception with symbolic execution, models can achieve verifiable arithmetic. A hybrid objective may minimise:

$$\mathcal{L}_{\text{hybrid}} = \mathcal{L}_{\text{neural}} + \gamma \|\text{Exec}(p_{\text{pred}}) - a_{\text{gold}}\|_2^2, \tag{2.67}$$

where $\text{Exec}(p_{\text{pred}})$ executes the generated symbolic program p_{pred}, a_{gold} is the gold numerical answer, and $\gamma > 0$ is a regularisation coefficient. Integrating differentiable calculators or constrained decoders could further enhance numerical faithfulness.

Challenge 4: Explainability and Faithful Reasoning. While large vision–language models achieve strong accuracy, their decision-making processes are opaque. Attention heatmaps only partially correlate with actual reasoning chains. Faithful reasoning requires explicit grounding between predicted answers and evidence cells.

Future systems should include explicit *rationale supervision*, evaluating correctness at each reasoning step (Eq. 2.41). Reinforcement-learning-based optimisation can promote faithful behaviour via reward shaping:

$$R = \alpha\, \text{Acc}_{\text{ans}} + \beta\, \text{IoU}_{\text{evidence}} + \delta\, \text{StepAcc}_{\text{reason}}, \tag{2.68}$$

where $\alpha, \beta, \delta > 0$ are weights and $\text{IoU}_{\text{evidence}}$ measures the spatial overlap between predicted and annotated evidence bounding boxes.

Challenge 5: Data Efficiency and Weak Supervision. Creating high-quality table annotations—especially at the cell or reasoning level—is expensive. Weakly supervised methods that leverage question–answer pairs without explicit cell labels represent an efficient alternative. Pseudo-labelling and expectation–maximisation can infer latent evidence:

$$\mathcal{L}_{\text{weak}} = \mathbb{E}_{\mathcal{S} \sim P(\mathcal{S}|q,a)}[-\log P(a \mid \mathcal{S}, q)], \tag{2.69}$$

where $P(\mathcal{S}|q, a)$ is the posterior distribution over supporting cell sets given the question–answer pair, updated iteratively from model predictions. Generative data augmentation using LLMs can further improve data diversity.

Challenge 6: Privacy and Sensitive Information. Financial and healthcare documents often contain confidential data. Training models on such data raises compliance and privacy concerns. Recent works explore *privacy-preserving table understanding* using differential privacy (DP) and federated learning (FL). A DP guarantee can be formalised as:

$$P(\mathcal{M}(D_1) \in S) \leq e^{\varepsilon} P(\mathcal{M}(D_2) \in S), \tag{2.70}$$

for any neighbouring datasets D_1, D_2 differing in one record and output set S (with $\delta = 0$ for pure DP). Integrating DP-SGD or federated aggregation (FedAvg) ensures sensitive table data remain local.

Challenge 7: Multi-modal Integration Beyond Tables. Real-world documents often include tables, charts, text, and diagrams. Future systems must perform holistic reasoning across these modalities. Joint modelling can be achieved via hierarchical attention:

$$\mathbf{h}_{\text{doc}} = \sum_m \alpha_m \mathbf{h}_m^{(\text{mod})}, \quad \alpha_m = \frac{\exp(w^\top \mathbf{h}_m^{(\text{mod})})}{\sum_{m'} \exp(w^\top \mathbf{h}_{m'}^{(\text{mod})})}, \tag{2.71}$$

where $\mathbf{h}_m^{(\text{mod})} \in \mathbb{R}^d$ denotes the embedding of modality m (e.g., table, chart, paragraph) and $w \in \mathbb{R}^d$ is a context vector. This architecture encourages dynamic fusion of heterogeneous visual structures.

Challenge 8: Evaluation Paradigm Reform. Current metrics reward output correctness but ignore reasoning validity. Next-generation evaluation frameworks should assess both outcome and process. Possible directions include:

- **Process-based evaluation:** Comparing model-generated reasoning traces with human logical steps.
- **Verifiable VQA:** Embedding symbolic executors to check consistency between intermediate operations and final answers.
- **Human-in-the-loop metrics:** Measuring trust and interpretability in professional domains such as finance or medicine.

Emerging Research Directions.

1. **Unified Table Foundation Models:** Developing large-scale pretraining frameworks that jointly handle table images, HTML tables, and spreadsheets within a single model.
2. **Reinforcement Learning for Structural Correction:** Using reward-driven policies to iteratively refine cell boundaries and structure predictions.
3. **Graph–Transformer Hybrids:** Combining graph message passing with transformer attention to encode fine-grained spatial relations.
4. **Self-verifiable Table VQA:** Incorporating symbolic execution and numerical simulators directly into vision–language decoders.

Conclusion. Table understanding has progressed from heuristic line-detection to multimodal foundation models capable of end-to-end reasoning. Yet, critical challenges remain—robust generalisation, faithful reasoning, privacy, and data efficiency. Addressing these will pave the way toward universally deployable systems for *Table Structure Recognition and Semantic Table Understanding through Table VQA*, realising the long-standing goal of interpretable document intelligence.

Chapter 3

TabStruct-Net: Anchor-Driven Table Structure Recognition

3.1 Introduction

Deep neural networks have shown promising results in understanding document layouts [10], [11], [130]. However, more needs to be done for structural and semantic understanding. Among these, the problem of table structure recognition has been of high interest in the community [3], [8], [9], [12], [13], [18], [24], [31], [41], [46], [48], [73], [131]–[135]. Table structure recognition refers to representation of a table in a machine-readable format, where its layout is encoded according to a pre-defined standard [3], [13], [18], [24], [131], [134]. It can be represented in the form of either physical [18], [24], [131], [134] or logical formats [3], [13]. While logical structure contains every cells' row and column spanning information, physical structure additionally contains bounding box coordinates. Table structure recognition is a precursor to contextual table understanding, which has a myriad of applications in business document analysis, information retrieval, visualization, and human-document interactions, as motivated in Figure 3.1.

Table structure recognition is a challenging problem due to complex structures and high variability in table layouts [3], [8], [9], [12], [13], [18], [24], [31], [46], [73], [131]–[134]. Early attempts in this space are dependent on extraction of hand-crafted features and meta-data extracted from the PDFs on top of heuristic/rule-based algorithms [19]–[21], [25] to locate tables and understanding tables by predicting/recognizing structures. These methods, however, fail to extend to scanned documents as they rely on meta-data information contained in the PDFs. They also make strong assumptions about the structure of the tables. Some of these methods are also dependent on textual information analysis which make them domain dependent. While textual features are useful, visual analysis becomes imperative for analysis of complex page objects. Inconsistency of size and density of tables, presence and location of table cell borders, variation in table cells' shapes and sizes, table cells spanning multiple rows and/or columns and multi-line content are some challenges (refer Figure 3.2 for some examples) that need to be addressed to solve the problem using visual cues [8], [9], [19]–[21], [25].

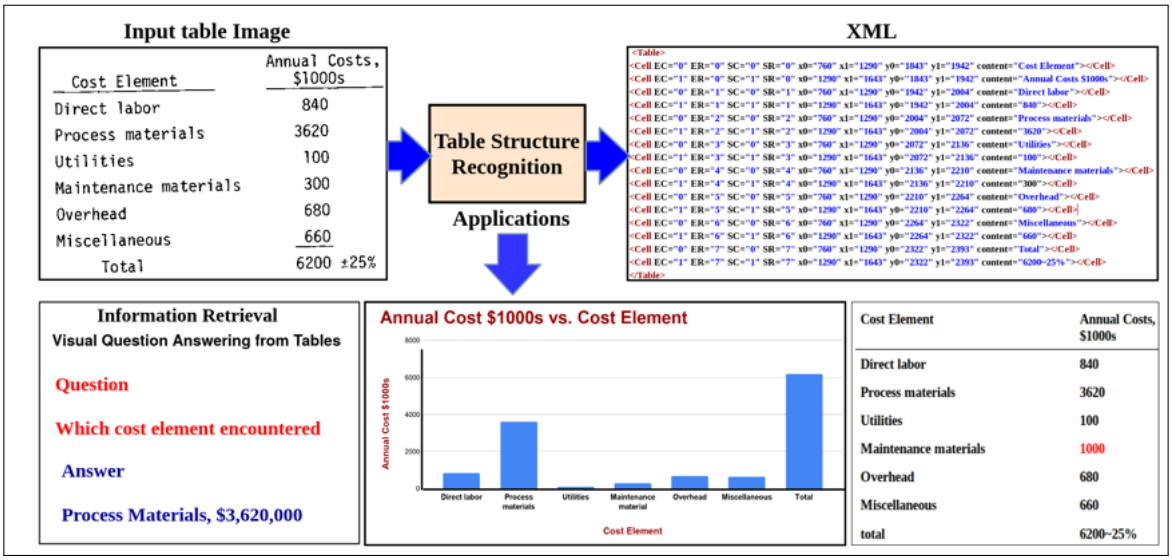
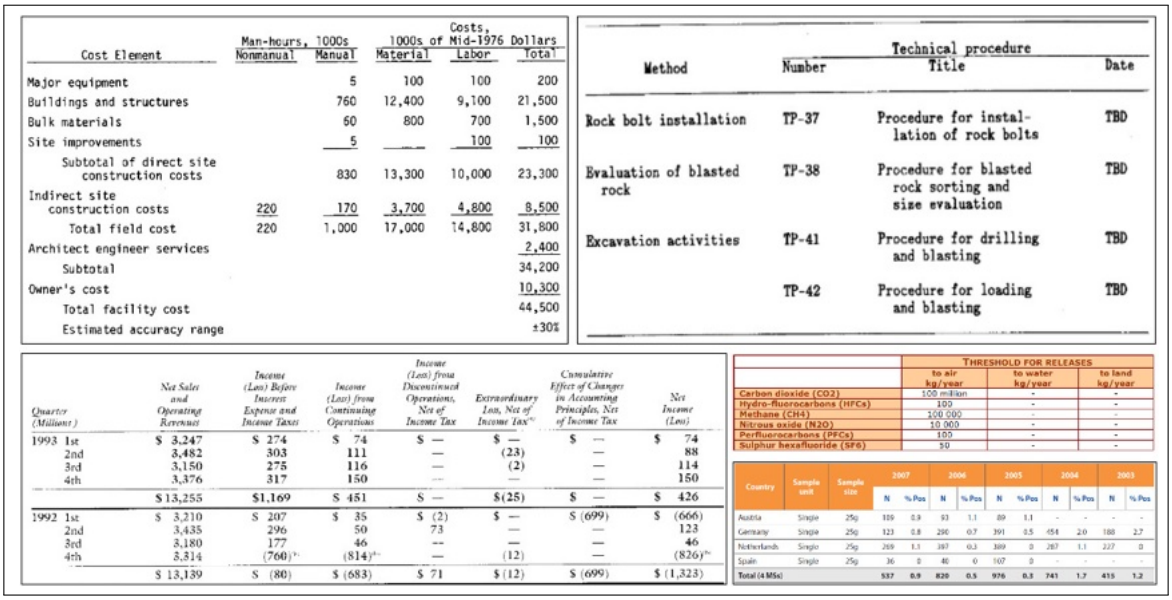


Figure 3.1: The figure depicts the problem of recognizing table structure from it's image. This opens up many applications including information retrieval, graphical representation and digitizing for editing.

We pose the table structure recognition problem as the generation of XML containing table's physical structure in terms of bounding boxes along with spanning information and, additionally, digitized content for every cell (see Figure 3.1). Since our method aims to predict this table structure given the table image only (without using any meta-information), we employ a two-step process — (a) *top-down:* where we decompose the table image into fundamental table objects, which are table cells using a cell detection network and (b) *bottom-up:* where we re-build the entire table as a collection of all the table cells localized from the top-down process, along with their row and column associations with every other cell. We represent row and column associations of table cells using row and column adjacency matrices.

Though table detection has observed significant success [5]–[7], [13], [136], detection of table cells remains a challenging problem. This is because of (i) large variation in sizes and aspect ratios of different cells present in the same table, (ii) cells' inherent alignment despite high variance in text amount and text justification, (iii) lack of linguistic context in cells' content, (iv) presence of empty cells and (v) presence of cells with multi-line content. To overcome these challenges, we introduce a novel loss function that models the inherent alignment of cells in the cell detection network; and a graph-based problem formulation to build associations between the detected cells. Moreover, as detection of cells and building associations between them depend highly on one another, we present a novel end-to-end trainable architecture, termed as TabStruct-Net, for cell detection and structure recognition. We evaluate our model for physical structure recognition on benchmark datasets: SciTSR [24], SciTSR-COMP [24], ICDAR-2013 table recognition [48], ICDAR-2019 (cTDaR) archival [41], and UNLV [137].



Cost Element	Man-hours, 1000s Nonmanual	Manual	Costs, 1000s of Mid-1976 Dollars Material	Labor	Total
Major equipment		5	100	100	200
Buildings and structures		760	12,400	9,100	21,500
Bulk materials		60	800	700	1,500
Site improvements		5		100	100
Subtotal of direct site construction costs		830	13,300	10,000	23,300
Indirect site construction costs	220	170	3,700	4,800	8,500
Total field cost	220	1,000	17,000	14,800	31,800
Architect engineer services					2,400
Subtotal					34,200
Owner's cost					10,300
Total facility cost					44,500
Estimated accuracy range					±30%

Method	Technical procedure Number	Title	Date
Rock bolt installation	TP-37	Procedure for installation of rock bolts	TBD
Evaluation of blasted rock	TP-38	Procedure for blasted rock sorting and size evaluation	TBD
Excavation activities	TP-41	Procedure for drilling and blasting	TBD
	TP-42	Procedure for loading and blasting	TBD

Quarter (Millions)	Net Sales and Operating Revenues	Income (Loss) Before Interest Expense and Income Taxes	Income (Loss) from Continuing Operations	Income (Loss) from Discontinued Operations, Net of Income Tax	Extraordinary Loss, Net of Income Tax[a]	Cumulative Effect of Changes in Accounting Principles, Net of Income Tax	Net Income (Loss)
1993 1st	$ 3,247	$ 274	$ 74	$ —	$ —	$ —	$ 74
2nd	3,482	303	111	—	(23)	—	88
3rd	3,150	275	116	—	(2)	—	114
4th	3,376	317	150	—	—	—	150
	$ 13,255	$1,169	$ 451	$ —	$(25)	$ —	$ 426
1992 1st	$ 3,210	$ 207	$ 35	$ (2)	$ —	$ (699)	$ (666)
2nd	3,435	296	50	73	—	—	123
3rd	3,180	177	46	—	—	—	46
4th	3,314	(760)[b]	(814)[b]	—	(12)	—	(826)[b]
	$ 13,139	$ (80)	$ (683)	$ 71	$ (12)	$ (699)	$ (1,323)

	THRESHOLD FOR RELEASES to air kg/year	to water kg/year	to land kg/year
Carbon dioxide (CO2)	100 million	-	-
Hydro-fluorocarbons (HFCs)	100	-	-
Methane (CH4)	100 000	-	-
Nitrous oxide (N2O)	10 000	-	-
Perfluorocarbons (PFCs)	100	-	-
Sulphur hexafluoride (SF6)	50	-	-

Country	Sample unit	Sample size	2007 N	2007 % Pos	2006 N	2006 % Pos	2005 N	2005 % Pos	2004 N	2004 % Pos	2003 N	2003 % Pos
Austria	Single	25g	109	0.9	93	1.1	89	1.1	-	-	-	-
Germany	Single	25g	123	0.8	290	0.7	391	0.5	458	2.0	188	2.7
Netherlands	Single	25g	269	1.1	397	0.3	389	0	287	1.1	227	0
Spain	Single	25g	36	0	40	0	107	0	-	-	-	-
Total (4 MSs)			537	0.9	820	0.5	976	0.3	741	1.7	415	1.2

Figure 3.2: Examples of complex table images from UNLV and ICDAR-2013 datasets. Complex tables are ones which contain partial or no ruling lines, multi-row/column spanning cells, multi-line content, many empty dense cells.

Further, we extend the comparative analysis of the proposed work for logical structure recognition on TableBank [13] dataset. Our method sets up a new direction for table structure recognition as a collaboration of cell detection, establishing an association between localized cells and, additionally, cells' content extraction.

3.2 Problem Definition and Challenges

Given an input table image $I \in \mathbb{R}^{H \times W \times 3}$, the objective of TSR is to predict a set of cell bounding boxes $\mathcal{B} = \{b_i\}_{i=1}^{N}$ and corresponding relational matrices $\mathbf{M}^{(r)}, \mathbf{M}^{(c)} \in \{0,1\}^{N \times N}$ indicating whether two cells belong to the same row or column:

$$\mathbf{M}_{ij}^{(r)} = \begin{cases} 1, & \text{if cells } i,j \text{ share a row,} \\ 0, & \text{otherwise,} \end{cases} \qquad \mathbf{M}_{ij}^{(c)} = \begin{cases} 1, & \text{if cells } i,j \text{ share a column,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

The reconstruction process finally yields the structured output

$$\mathcal{S} = \{(b_i, SR_i, ER_i, SC_i, EC_i)\}_{i=1}^{N},$$

where SR, ER, SC, EC denote start- and end-indices for rows and columns.

3.2.1 Complexities of Table Perception

The main difficulties arise from the tension between visual irregularity and structural regularity:

- **Diverse Topologies:** Tables may contain nested headers, hierarchical columns, or missing ruling lines, producing non-rectangular partitions.
- **Multi-Span Cells:** A single cell may extend across multiple rows or columns, violating one-to-one correspondences between grid lines and visual cues.
- **Empty or Semi-Empty Cells:** Blank regions often encode semantic information (e.g., "not applicable"), demanding detection without text evidence.
- **Noise and Degradation:** Scanned or historical documents introduce skew, bleed-through, and uneven illumination that confound edge-based methods.

3.2.2 Our Contributions

We address these challenges by formulating table structure recognition as a two-stage process that combines top-down and bottom-up visual cues. In the first stage, the table image is decomposed into its fundamental components—table cells—using an object detection approach (the top-down step). In the second stage, the table is reconstructed by establishing associations between the detected cells to infer the overall structure (the bottom-up step). More specifically, the output of our system is an XML representation of the table that contains each cell's bounding box along with its row and column spanning information, and optionally the textual content of each cell. This concept is illustrated in Figure 3.1, which shows how an input table image can be converted into a structured representation usable for downstream applications.

While detecting entire tables in documents is a well-studied problem [5]–[7], [13], [136], detecting individual table cells is considerably more challenging. Several of the challenges listed above directly affect cell detection. For instance, cells in the same row or column are expected to be perfectly aligned in the final structure, yet the visual content (text) within each cell may not line up exactly due to varying text lengths and alignments. Likewise, empty cells provide no text or clear visual markers, making them easy to miss. To tackle these issues, our approach introduces a novel alignment loss that explicitly enforces the structural alignment constraints during the cell detection training. In parallel, we adopt a graph-based formulation for the structure recognition stage to naturally model the relationships between cells once they are detected. Furthermore, because the success of the structure recognition stage depends on the quality of cell detection (and vice versa), we propose a unified end-to-end trainable architecture—called TabStruct-Net—that jointly optimizes both tasks. By training the cell detector and the structure recognizer together, TabStruct-Net allows feedback between the two tasks, leading to improved overall performance.

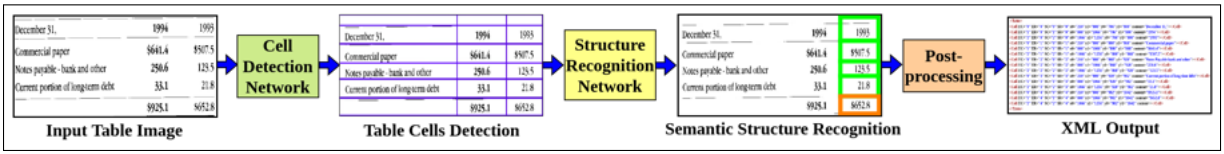


Figure 3.3: Block diagram of TabStruct-Net. We assume that the table has been detected and cropped from the document (table detection is a precursor step to structure recognition). The end-to-end TabStruct-Net architecture then predicts all cell bounding boxes and the relationships between cells in a single forward pass. Finally, a post-processing heuristic uses the detected cells and their predicted associations to produce an XML output encoding the table.

We empirically evaluate the proposed TabStruct-Net on a wide range of public datasets that include both modern digital tables and complex archival documents. For physical structure recognition (i.e., predicting explicit cell locations and spans), we use the ICDAR-2013 table competition dataset [48], the ICDAR-2019 cTDaR archival dataset [41], the UNLV dataset [137], and the SciTSR dataset (including its "complicated" subset) introduced by Chi et al. [24]. To further demonstrate the generality of our approach, we also evaluate on the large synthetic TableBank dataset [13] and the PubTabNet dataset [3] for logical structure recognition, where the goal is to produce a structured representation (e.g., HTML or LaTeX) of the table without explicit coordinates. Our method achieves state-of-the-art results across multiple benchmarks and is flexible enough to handle both digitally-born tables (with clear structure and content) and challenging scanned or handwritten tables (with irregular layouts).

Our main contributions in this chapter are summarized as follows:

- We demonstrate how combining top-down (cell detection) and bottom-up (structure recognition) visual cues enables reliable table structure recognition in document images. We present an end-to-end trainable architecture, TabStruct-Net, that jointly learns table cell detection and structure recognition in one unified network.

- We introduce a novel alignment loss function to enforce structural constraints (cell alignment in rows and columns) during cell detection training, and we modify the Feature Pyramid Network (FPN) in the detector for better capture of low-level and long-range visual features.

- We enhance the visual feature representation for structure recognition by incorporating an LSTM-based sequential encoding of each cell's region, building on the graph-based model of Qasim et al. [12].

- We consolidate and compare results from numerous existing methods on standard benchmarks, providing a thorough comparative study of table structure recognition approaches.

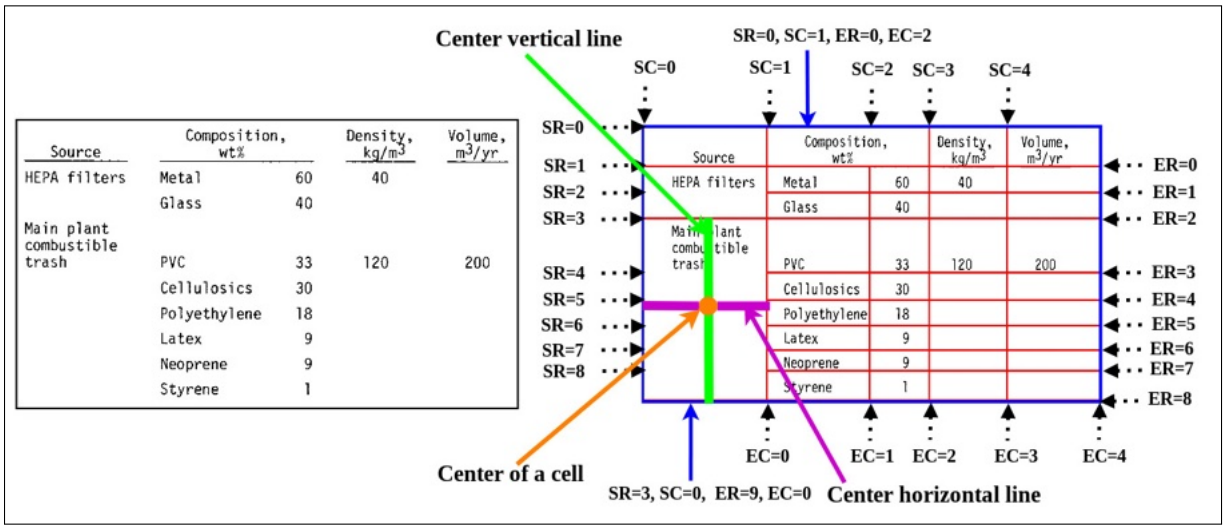


Figure 3.4: Illustration of cell spanning information using a table example from the UNLV dataset. **Left:** an example table image. **Right:** the same table with each cell annotated by its row and column spans (start-row, end-row, start-col, end-col indices). Cells that occupy multiple rows or columns are indicated by spanning across those indices (colored bands).

3.3 TabStruct-Net

Our TabStruct-Net approach performs table structure recognition in three main steps (Figure 3.3): (a) detection of table cells, (b) establishing row/column relationships between the detected cells, and (c) post-processing to produce a structured XML output. We describe each of these components in detail, including the network architecture, training procedures, and techniques introduced to address the challenges discussed earlier.

3.3.1 Top-Down: Cell Detection

The first stage of TabStruct-Net is a table cell detection network, which takes a table image and predicts bounding boxes for all cells in the table. We adopt the general object detection paradigm for this task, building upon the success of region-based convolutional detectors like Faster R-CNN and Mask R-CNN in natural scene images [29], [30], [138]. In our case, each table cell is treated as an object to be detected. However, detecting table cells differs from typical object detection because table cells have an inherent alignment and grid-like arrangement relative to each other (something not true for generic objects in a scene). This suggests that incorporating knowledge of the overall table structure can aid in better cell localization. We use the Mask R-CNN framework [29] as our base detector due to its strong performance in instance segmentation and its built-in Region Proposal Network (RPN) for object proposals. We introduce several enhancements to the standard Mask R-CNN to tailor it for table cell detection:

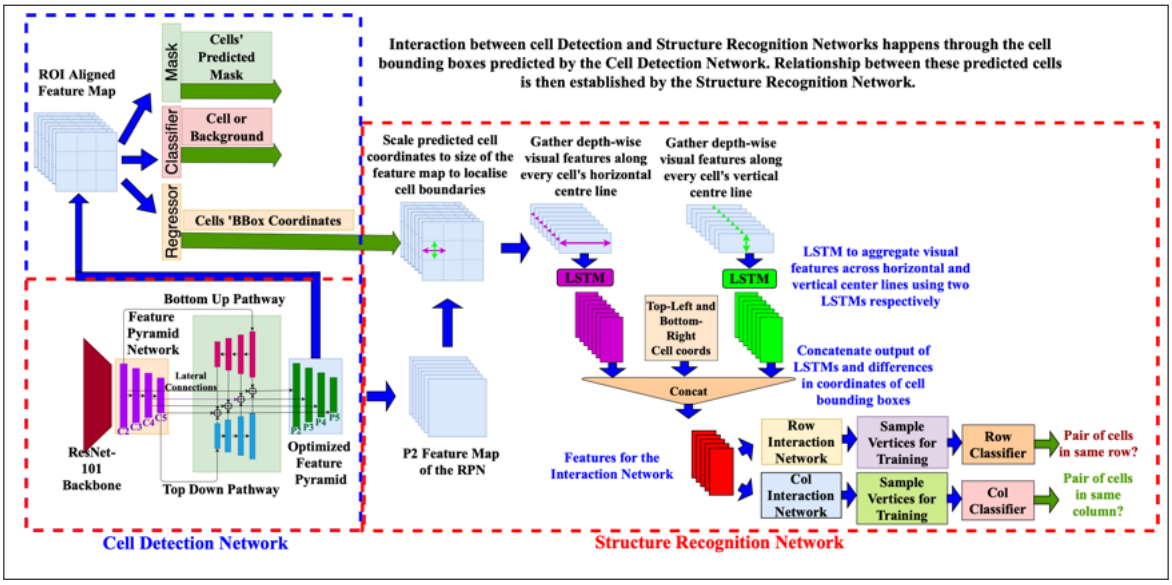


Figure 3.5: Our TabStruct-Net. Modified RPN in cell detection network, which consists of both top-down and bottom-up pathways to better capture low-level visual features. P2 layer of the optimized feature pyramid is used in the structure recognition network to extract visual features.

- **Dilated RPN Convolutions:** We augment the RPN with dilated convolutional layers [139], [140] to capture long-range horizontal and vertical context in the table image. Standard Mask R-CNN employs 3×3 convolutions within its RPN. To capture the quasi-grid layout of tables, TabStruct-Net introduces *dilated* convolutions [139] with dilation factor $d = 2$:

$$y(p) = \sum_{k \in \Omega} w_k \, x(p + d \cdot k), \tag{3.2}$$

  which enlarges the receptive field without loss of resolution, allowing the RPN to perceive distant horizontal or vertical separators. This enhancement significantly improves recall for multi-span cells and dense column structures.

- **Augmented FPN with Top-Down Path:** We modify the Feature Pyramid Network (FPN) in Mask R-CNN by adding an extra top-down pathway [141] that propagates high-level semantic feature maps down to the lower-resolution feature maps. In effect, the FPN is made bi-directional (with the usual bottom-up feature hierarchy plus a top-down refinement). This helps the detector handle cells at varying scales: small cells benefit from high-level context, while large cells still utilize low-level detail, improving detection consistency across different cell sizes. A bidirectional

FPN (Bi-FPN) integrates top-down and bottom-up pathways:

$$F_l^{\text{td}} = \text{Conv}\big(\alpha_l\, F_{l+1}^{\text{td}} + (1-\alpha_l)\, F_l\big), \tag{3.3}$$

$$F_l^{\text{bu}} = \text{Conv}\big(\beta_l\, F_{l-1}^{\text{bu}} + (1-\beta_l)\, F_l^{\text{td}}\big), \tag{3.4}$$

where α_l, β_l are learnable fusion weights normalized via softmax across levels. The Bi-FPN ensures both fine-grained edge details and global semantic cues are available to downstream modules.

- **Anchor Configuration and Detection Loss:** Anchor boxes parameterize candidate regions. Let $A = \{a_k\}_{k=1}^{K}$ denote anchors with scales $s_k \in \{8, 16, 32, 64, 128\}$ and aspect ratios $r \in \{0.5, 1, 2\}$. Each anchor generates regression targets

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \quad t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$

yielding the standard multi-task RPN loss:

$$L_{\text{RPN}} = \frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}}(p_i, p_i^*) + \lambda \frac{1}{N_{\text{reg}}} \sum_i p_i^* L_{\text{reg}}(t_i, t_i^*), \tag{3.5}$$

where p_i and p_i^* denote predicted and ground-truth objectness scores, respectively. λ balances classification and regression terms.

The detector further includes Mask R-CNN losses $L_{\text{box}}, L_{\text{cls}}, L_{\text{mask}}$. Collectively they compose the detection loss:

$$L_{\text{det}} = L_{\text{RPN}} + L_{\text{box}} + L_{\text{cls}} + L_{\text{mask}}. \tag{3.6}$$

- **Structural Alignment Loss:** In addition to the standard object detection losses (classification and bounding-box regression, and mask segmentation loss if applicable), we introduce additional loss terms during training to encode the inherent alignment constraints between table cells. This drives the detector to predict cell boxes that align neatly in rows and columns. We incorporate these constraints in two ways: (i) by training the cell detection network jointly with the structure recognition network (so that feedback from structure recognition can influence cell detection, and (ii) by defining a novel explicit alignment loss function that penalizes misalignment of predicted cells.

3.3.1.1 Alignment Loss: Derivation and Analysis

Although Eq. (3.6) ensures local accuracy, independent regression per cell yields spatial inconsistencies across rows and columns. To enforce structural coherence, TabStruct-Net introduces the **alignment loss** L_{align}, formulated over pairs of cells predicted to share alignment relationships. We leverage the fact that, in a properly structured table, any two cells that lie in the same row should start at the same

vertical position and end at the same vertical position (assuming a consistent top and bottom alignment for that row). Similarly, cells in the same column should share the same left (start) and right (end) horizontal positions. To model these constraints, we first preprocess the ground truth cell bounding boxes to ensure perfect alignment: if the provided annotations are at the content level (tight around text) as is the case in some datasets [3], [24], [48], we expand or adjust each box such that all cells in the same row have an identical top and bottom boundary, and all cells in the same column share the same left and right boundaries. This yields a set of aligned ground truth coordinates for training.

Formulation For each row r containing cells $C_r = \{c_i\}$ and each column c containing cells C_c,

$$L_y^{(1)} = \frac{1}{|R|} \sum_r \frac{1}{|C_r|^2} \sum_{i,j \in C_r} (y_i^1 - y_j^1)^2, \tag{3.7}$$

$$L_y^{(2)} = \frac{1}{|R|} \sum_r \frac{1}{|C_r|^2} \sum_{i,j \in C_r} (y_i^2 - y_j^2)^2, \tag{3.8}$$

$$L_x^{(1)} = \frac{1}{|C|} \sum_c \frac{1}{|C_c|^2} \sum_{i,j \in C_c} (x_i^1 - x_j^1)^2, \tag{3.9}$$

$$L_x^{(2)} = \frac{1}{|C|} \sum_c \frac{1}{|C_c|^2} \sum_{i,j \in C_c} (x_i^2 - x_j^2)^2. \tag{3.10}$$

The complete regularizer is

$$L_{\text{align}} = L_y^{(1)} + L_y^{(2)} + L_x^{(1)} + L_x^{(2)}. \tag{3.11}$$

This formulation minimizes intra-row and intra-column coordinate variance, effectively projecting all aligned cells toward shared baselines.

Gradient Interpretation Let $\theta_i = (x_i^1, x_i^2, y_i^1, y_i^2)$ be the predicted parameters of cell i. Differentiating Eq. (3.11) yields for each coordinate:

$$\frac{\partial L_{\text{align}}}{\partial y_i^1} = \frac{2}{|R|} \sum_{r: i \in C_r} \frac{1}{|C_r|^2} \sum_{j \in C_r} (y_i^1 - y_j^1),$$

and analogously for y_i^2, x_i^1, x_i^2. This gradient term attracts misaligned boundaries toward the centroid of their peers, akin to Laplacian smoothing on spatial coordinates. Unlike standard box regression, which depends only on ground-truth correspondence, L_{align} couples predictions across cells, enforcing relational geometry during back-propagation.

Connection to Structural Priors The alignment loss can be interpreted as minimizing an energy function

$$E(\Theta) = \frac{1}{2} \Theta^\top \mathbf{L} \Theta, \tag{3.12}$$

where $\mathbf{L}$ is a graph Laplacian defined over row- and column-wise neighborhoods. Minimizing E enforces smooth coordinate fields consistent with the grid prior. This insight connects TabStruct-Net to manifold regularization frameworks, embedding geometric consistency directly into deep detection.

3.3.2 Structure Recognition

The alignment-regularized detector outputs a set of candidate cell boxes $\mathcal{B} = \{b_i\}_{i=1}^{N}$, each accompanied by a region-of-interest feature vector $\mathbf{f}_i \in \mathbb{R}^d$ extracted from the P_2 layer of the Bi-FPN. To infer structural relations among cells, TabStruct-Net introduces a *Dynamic Graph Convolutional Neural Network (DGCNN)* [12]. In our formulation, each detected table cell corresponds to a vertex in a graph. We then define two adjacency matrices: M_{row} for rows and M_{col} for columns. Both M_{row} and M_{col} are $N \times N$ binary matrices (where N is the number of detected cells). The entry $M_{row}[i, j] = 1$ if cell i and cell j belong to the same row, and 0 otherwise; similarly $M_{col}[i, j] = 1$ if cells i and j are in the same column. By construction, these adjacency matrices are symmetric. The goal of the structure recognition network is to predict these two matrices, given the image and the set of cell detections.

During training, we prepare target row and column adjacency matrices from the ground truth table structure (i.e., using the known row/column indices of each ground-truth cell). During inference, the network will output predicted adjacency matrices. In training, we include a detected cell in the structure recognition stage only if it significantly overlaps a ground-truth cell (Intersection-over-Union IoU ≥ 0.5) to focus the structure learning on correctly detected cells and ignore spurious detections. At test time, however, the structure recognizer will consider all detected cells (with no ground truth to filter by).

Our structure recognition network has three components, which together compute features for each cell and then classify pairs of cells as same-row or same-column:

- **Visual Feature Component:** We extract a visual feature vector for each detected cell that captures its appearance and spatial extent. To do so, we take the feature map from the P2 layer of the detector's FPN (the finest resolution feature map) and sample features from it for each cell's region. Instead of using a single feature vector at the cell's centroid as in some prior work [12], we want to encode the cell's full span. We therefore sample features along the cell's central horizontal line and central vertical line (effectively scanning across the cell's width and height) and feed these sequences of features into two orthogonal LSTM encoders (one horizontal, one vertical). The LSTM (long short-term memory) networks [142] aggregate the visual information across the cell's interior in both directions, producing a fixed-length representation that reflects the cell's visual content and shape across its full extent. This yields a visual feature vector for each cell that is sensitive to attributes like the cell's background shading, border lines, or presence of text across the whole cell region.

- **Interaction Component:** We then model the interactions between cells using a Graph Neural Network. We adopt a dynamic graph CNN (DGCNN) architecture similar to the one used by Qasim et al. [12] for table parsing. In this component, each cell (node) begins with an initial feature (e.g., the visual feature from the LSTM plus geometric features like the cell's bounding box coordinates). The graph network then iteratively updates each cell's representation by aggregating information from its neighboring cells. We define the graph in terms of spatial proximity: each cell considers a certain number of nearest neighbor cells as potential connections. Given vertex features $\{\mathbf{f}_i\}$, pairwise neighborhoods are defined in the current feature space. For each node i, its k-nearest neighbors $\mathcal{N}_k(i)$ are selected using Euclidean distance:

$$\mathcal{N}_k(i) = \arg\operatorname{top}_k\big(-\|\mathbf{f}_i - \mathbf{f}_j\|_2\big).$$

Edges are directed from i to each $j \in \mathcal{N}_k(i)$, forming a dynamic graph $G_t = (V, E_t)$ that evolves with layer depth t. Specifically, we connect each cell to its k closest cells (by Euclidean distance between cell centroids in the image) and use an edge convolution (EdgeConv) operation as in DGCNN [143] to compute interaction features. The GNN effectively spreads information about the layout: for instance, a cell will receive signals from the cells above and below it (likely in the same column) and from the cells to its left and right (likely in the same row), allowing it to infer its structural context. For every edge (i, j), an edge feature is computed as

$$\mathbf{e}_{ij} = \phi_\theta\big([\mathbf{f}_i,\, \mathbf{f}_j - \mathbf{f}_i,\, \Delta x_{ij}, \Delta y_{ij},\, |\Delta w_{ij}|, |\Delta h_{ij}|]\big), \tag{3.13}$$

where ϕ_θ is a shared MLP and the geometric terms $\Delta x_{ij}, \Delta y_{ij}, \Delta w_{ij}, \Delta h_{ij}$ encode relative position and scale. The vertex update rule is then

$$\mathbf{f}'_i = \max_{j \in \mathcal{N}_k(i)} \mathbf{e}_{ij}, \tag{3.14}$$

using max-pooling as an order-invariant aggregator.

Multiple DGCNN layers progressively refine vertex embeddings $\mathbf{f}_i^{(t)}$.

- **Classification Component:** After one or more rounds of message passing in the interaction GNN, we obtain a refined feature vector (interaction feature) for each cell that encodes information about its neighbors. To actually predict adjacency (row/column association) between any two cells, we take the pair of cells in question, concatenate their interaction feature vectors, and also include some explicit geometric features (such as the difference in their bounding box coordinates). This combined feature is then fed to two binary classifiers: one classifier outputs whether the two cells belong to the same row, and another whether they belong to the same column. We use a Monte Carlo sampling strategy during training to select balanced pairs of cells to feed to the classifiers (following the approach of Qasim et al. [12]), whereas at test time we evaluate the classifiers on every pair of detected cells to construct the full adjacency matrices. MLP classifier to predict

adjacency scores for rows and columns is defined as:

$$p_{ij}^{(r)} = \sigma(\mathbf{w}^{(r)\top}[\mathbf{f}_i, \mathbf{f}_j]), \tag{3.15}$$

$$p_{ij}^{(c)} = \sigma(\mathbf{w}^{(c)\top}[\mathbf{f}_i, \mathbf{f}_j]), \tag{3.16}$$

where $\sigma(\cdot)$ denotes the logistic sigmoid.

3.3.2.1 Graph-Based Loss Formulation

Each pairwise prediction is supervised by binary cross-entropy:

$$L_{\text{row}} = -\frac{1}{N^2}\sum_{i,j}\big[M_{ij}^{(r)}\log p_{ij}^{(r)} + (1 - M_{ij}^{(r)})\log(1 - p_{ij}^{(r)})\big], \tag{3.17}$$

$$L_{\text{col}} = -\frac{1}{N^2}\sum_{i,j}\big[M_{ij}^{(c)}\log p_{ij}^{(c)} + (1 - M_{ij}^{(c)})\log(1 - p_{ij}^{(c)})\big]. \tag{3.18}$$

The combined structural loss is

$$L_{\text{gnn}} = \lambda_r L_{\text{row}} + \lambda_c L_{\text{col}}, \tag{3.19}$$

with $\lambda_r = \lambda_c = 0.5$ by default. The balance ensures equal emphasis on both directional adjacencies.

The gradient of L_{gnn} w.r.t. node features $\mathbf{f}_i$ couples all cells through their predicted adjacency, promoting feature smoothness along structurally consistent edges—a property that empirically enhances detection confidence.

3.3.3 Joint End-to-End Optimization

All components are trained jointly by minimizing

$$L_{\text{total}} = L_{\text{det}} + \gamma_1 L_{\text{align}} + \gamma_2 L_{\text{gnn}}, \tag{3.20}$$

where $\gamma_1 = 1.0$ and $\gamma_2 = 0.5$ were empirically chosen. Joint training encourages mutual reinforcement: better-aligned boxes improve adjacency classification, and the GNN's structural gradients guide the detector toward spatially coherent layouts.

3.3.4 Post-Processing and XML Reconstruction

After inference, detected boxes b_i and predicted adjacency matrices $\mathbf{M}^{(r)}, \mathbf{M}^{(c)}$ are combined to infer grid coordinates. For each connected row component R_k and column component C_l, row and

column indices are assigned as

$$SR_i = \min_{j:M_{ij}^{(r)}=1} R_j, \quad ER_i = \max_{j:M_{ij}^{(r)}=1} R_j, \tag{3.21}$$

$$SC_i = \min_{j:M_{ij}^{(c)}=1} C_j, \quad EC_i = \max_{j:M_{ij}^{(c)}=1} C_j. \tag{3.22}$$

In summary, the post-processing works as follows:

- **Row grouping:** We first sort all detected cells by their top y-coordinate (ascending). We then iterate through the sorted cells and group them into rows. The topmost cell defines row 0. We assign it row index 0, and then assign every other cell that is connected to it in the predicted row adjacency matrix M_{row} the same row index. Once all connected cells (forming one physical row) are labeled, we move to the next unlabeled cell in the sorted order, increment the row index, and repeat. This effectively finds all distinct sets of cells that belong to the same row.
- **Row span calculation:** Because the predicted M_{row} might link a cell to others not immediately adjacent (for example, spanning multiple rows), each cell can end up with a list of row indices it belongs to. We define each cell's start-row (SR) as the minimum row index in its group and its end-row (ER) as the maximum index in that list. If a cell is only in one row group, $SR = ER$ for that cell.
- **Column grouping:** We perform a similar procedure for columns: sort cells by left x-coordinate, group connected components according to M_{col} to assign column indices incrementally.
- **Column span calculation:** Each cell's start-column (SC) is set to the minimum column index assigned to that cell, and end-column (EC) to the maximum, analogous to the row case.

This step effectively merges overlapping spans, producing (SR, ER, SC, EC) tuples for each cell. The final structured output is serialized as:

```
<cell id="i"
 x1="..." y1="..." x2="..." y2="..."
 start-row="SR_i" end-row="ER_i"
 start-col="SC_i" end-col="EC_i"/>
```

Such XML representation ensures compatibility with evaluation benchmarks and facilitates downstream semantic parsing.

3.4 Experiments

3.4.1 Datasets

We use various benchmark datasets - SciTSR [24], SciTSR-COMP [24], ICDAR-2013 table recognition [48], ICDAR-2019 cTDaR archival [41], UNLV [137], Marmot extended [131], TableBank [13] and

PubTabNet [3] datasets for extracting structure information of tables. Statistics of these datasets are listed in Table 3.1.

Dataset	Train	Test
SciTSR	12000	3000
SciTSR-COMP	12000	716
ICDAR 2013	-	158
ICDAR-2013-PARTIAL	124	34
ICDAR 2019	600	150
UNLV	-	558
UNLV-PARTIAL	446	112
Marmot-extended	1016	-
TableBank	145K	1000
PubTabNet	339K	114K

Table 3.1: Statistics of the datasets used for our experiments (Train and Test split).

Our TabStruct-Net makes an assumption that all cells belonging to the same column are aligned with respect to x coordinates and cells belonging to the same row are aligned with respect to y coordinates. SciTSR [24], SciTSR-COMP [24] and ICDAR-2013 [48] datasets have ground truth bounding boxes at the level of cell's content (box is the smallest rectangular block that encapsulates the cell's content). To handle this, we expand the bounding boxes of every cell in a row and column to get maximum sized content-level box in a particular row and column.

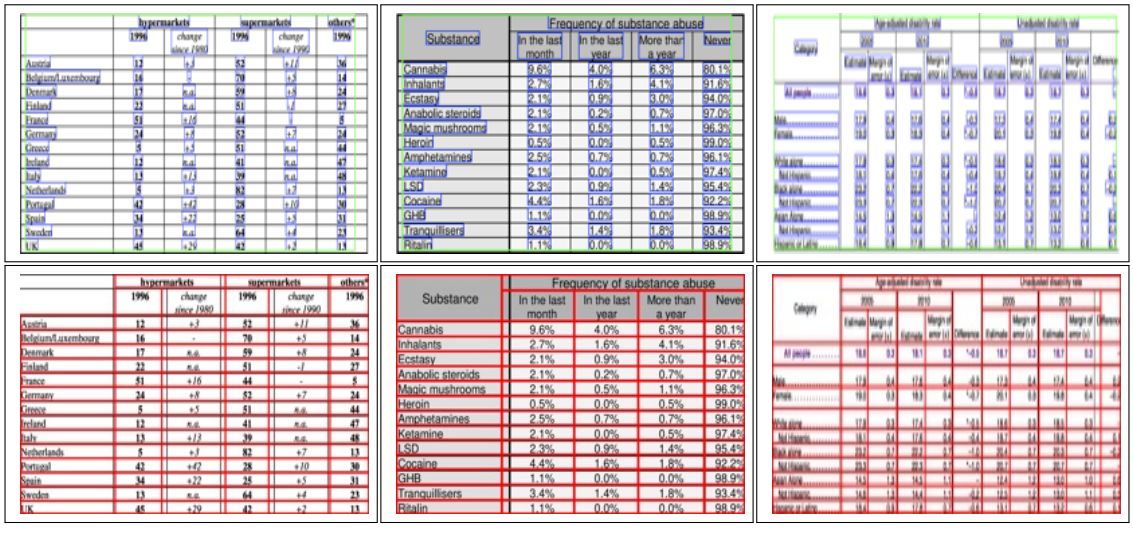


Figure 3.6: Ground truth unification. content-level bounding boxes are given in ground truth as shown in **First Row**. We make content-level bounding boxes into cell-level bounding boxes as shown in **Second Row**.

3.4.2 Baseline Methods

We compare the performance of our TabStruct-Net against seven following benchmark methods.

DeepDeSRT [46]: This method leverages semantic segmentation approach to localize each row and column from the table image. This method outputs table as a grid-like structure and fails to identify multiple row and multiple column spanning cells. Since no code is available, we implement our own version of this method. Since this method works by predicting every row and column, the SciTSR training dataset is pre-processed to obtain row and column level coordinates before training.

TableNet [131]: This method uses semantic segmentation approach to extract table and column masks, and segments rows by identifying words present in different columns (extracted using Tesseract OCR [45]) that occur at the same horizontal level. For comparison against other methods, we directly use the results reported by the authors.

GraphTSR [24]: This method consists of edge-to-vertex and vertex-to-edge graph attention blocks to compute vertex and edge representations in a latent space, and finally classify each edge as 'horizontal', 'vertical' or 'no-relation'. It uses absolute and relative positions of every cell extracted from the PDF to compute initial vertex and edge features.

SPLERGE [18]: This method comprises of two deep learning networks that perform splitting and merging operations in sequence to predict fine grid-like table structure and to predict merged cells which span multiple rows/columns. Split method shows an improved performance when additional PDF extracted meta-features are provided along with the table image. For the split model (to obtain the basic grid of the table), authors pre-process the ground truth by maximizing the row and column separator regions without intersecting any non multiple row or column spanning cell. For the merge model (to identify cells that span multiple rows or columns), the authors prepare the ground truth by identifying grid elements that span multiple cells and set the merging probability in the respective directions. Further, for evaluating this method on ICDAR-2013 [48] dataset, the authors realized that merge method did not work with a good performance, and hence, introduced the following specific heuristics to merge cells instead:

- Merge cells where separators pass through text.
- Merge adjacent blank columns with cells that have been formed by merging many cells.
- Merge adjacent blank rows with content cells.
- Split columns that have a consistent white-space gap between vertically aligned text.

DGCNN* [12]: Authors formulate the problem as a graph learning problem to predict whether every pair of words belongs to the same cell, row and/or column or not. Their architecture consists of a visual network, an interaction network and a classification network. For evaluation purposes, table image along with word-level bounding boxes is provided as inputs.

Bi-directional GRU [132]: Given the table image, this method uses two bi-directional GRUs to establish row and column boundaries in a context driven manner. This method however fails to localize multiple row and/or multiple column spanning cells.

Image-to-Text [131]: This method utilizes an Image-to-Markup model to predict a markup-like output of a given table image. It consists of a CNN based encoder to compute visual features and an LSTM based decoder to produce markup output.

3.4.3 Implementation Details

TabStruct-Net[1] model has been trained and evaluated with table images scaled to a fixed size of 1536×1536 while maintaining the original aspect ratio as the input. While training, cell-level bounding boxes along with row and column adjacency matrices (prepared from start-row, start-column, end-row and end-column indices) are used as the ground truth. We use NVIDIA TITAN X GPU with 12 GB memory for our experiments and a batch-size of 1. Instead of using 3×3 convolution on the output feature maps from the FPN, we use a dilated convolution with filter size of 2×2 and dilation parameter of 2. Also, we use the ResNet-101 backbone that is pre-trained on MS-COCO [45] dataset. To compute region proposals, we use 0.5, 1 and 2 as the anchor scale and anchor box sizes of 8, 16, 32, 64 and 128. Further, for generation of the row/column adjacency matrices, we use 2400 as the maximum number of vertices keeping in mind dense tables. Since every input table may contain hundreds of table cells, training can be a time consuming process.

To achieve faster training, we employ a 2-stage training process. In the first step, we use 2014 anchors and 512 RoIs. With this setting, the model is able to learn high and low level features but resulted in a large number of false negatives. To combat this, network is trained with 3072 anchors and 2048 RoIs. This significantly reduces the number of false negatives. For the first step, we train a total of 30 epochs, for the first 8, we train all FPN and subsequent layers, for the next 15, we train FPN + last 4 layers of ResNet-101 and for the last 7 epochs, we train all the layers of the model. For the second step, we train a total of 10 epochs, for the first 3, we train all FPN and subsequent layers, for the next 4, we train FPN + last 4 layers of ResNet-101 and for the last 3 epochs, we train all the layers of the model. During both the stages, we use 0.001 as the learning rate, 0.9 as the momentum and 0.0001 as the weight decay regularisation.

[1]Our code is available at `https://github.com/sachinraja13/TabStructNet.git`

Test Dataset	Train Dataset	S-A			S-B		
		P↑	R↑	F1↑	P↑	R↑	F1↑
ICDAR-2013	SciTSR	0.915	0.897	0.906	0.976	0.985	0.981
ICDAR-2013-partial	SciTSR	0.930	0.908	0.919	0.991	0.993	0.992
SciTSR	SciTSR	0.927	0.913	0.920	0.989	0.993	0.991
SciTSR-comp	SciTSR	0.909	0.882	0.895	0.981	0.987	0.984
UNLV-partial	SciTSR	0.849	0.828	0.839	0.992	0.994	0.993
ICDAR-2019	SciTSR	0.595	0.572	0.583	0.924	0.899	0.911
ICDAR-2019	ICDAR-2019	0.803	0.768	0.785	0.975	0.957	0.966
ICDAR-2019	SciTSR+ICDAR-2019	0.822	0.787	0.804	0.975	0.958	0.966

Table 3.2: shows the performance of our TabStruct-Net for physical table structure recognition on various benchmark datasets.

Test Dataset	Train Dataset	Metric	Score
TableBank-Word	SciTSR	BLEU	0.914
TableBank-LaTeX	SciTSR	BLEU	0.937
TableBank-Word+LaTeX	SciTSR	BLEU	0.916
PubTabNet	SciTSR	TEDS	0.901

Table 3.3: shows the performance of our TabStruct-Net for logical table structure recognition on various benchmark datasets.

3.4.4 Evaluation Measures

Details of Evaluation Criteria: For comparison against most of the existing methods, we use the precision, recall and F1 score [24], [48], [137]. Before evaluating performance of structure recognition, it is important to understand the specific cases in which detected cells are taken into consideration for structure recognition:

- We consider a detected cell to be a true positive if it overlaps with the ground truth cell bounding box is more than 0.5.
- During evaluation of structure recognition, cells that have no content (i.e., empty or blank cells) are discarded. It means that adjacency relations that involve a blank cell are not taken into consideration.

To evaluate the performance of structure recognition, adjacency relations between every cell (with content) are generated with their horizontal and vertical neighbors. This predicted relation list is then compared with the ground truth list to generate precision, recall and F1 measures.

As per [48], this method accounts for the standard evaluation measures for table structure recognition for the following reasons:

- It provides for a simple way to account for errors in the scenarios of complex table layouts containing blank cells, and cells that span multiple rows and/or columns.

- It accounts for evaluation of both physical as well as logical structure prediction methods as it is not dependent on the bounding box coordinates information.

3.4.5 Experimental Setup

One major challenge in the comparison study with the existing methods is the inconsistent use of additional information (e.g., meta-features extracted from the PDFs [18], content-level bounding boxes from ground truths [24], [131] and cell's location features generated from synthetic dataset [12]).

For the unification of fair comparison for table structure recognition, we divide all inconsistencies into several levels - (i) inconsistency with respect to input modalities, (ii) inconsistency with respect to annotation levels, (iii) inconsistency with respect to representation of table structure, (iv) inconsistency with respect to evaluation methods, and (v) inconsistency with respect to way of computing evaluation scores.

Inconsistency with respect to input modalities: Section 3.2 describes that many methods for table structure recognition work with table images alone [18], [46], [132], several other assume additional information in the form of meta-features or bounding boxes around every word or cell-content extracted from the PDFs [12], [24], [131]. This makes it difficult to compare these methods under a unified scenario. To take care of this problem, we define two different experimental setups - (a) **Setup-A (S-A)** where only table image is used as an input to the structure recognition model and (b) **Setup-B (S-B)** where table image along with additional meta-features such as low-level content bounding boxes are used as an input to the structure recognition model. We present our results under both the experimental setups for a thorough comparison of our work against most of the recent methods in this space. To achieve this, we train our model for cell detection as well as structure recognition collectively for S-A. For evaluation in S-B, instead of predicting cell bounding boxes from the image, we use the table image and the low-level bounding box information from OCR or ground truth to be able to directly and fairly compare our method.

Inconsistency with respect to annotation levels: It is important to note that training of TabStructNet assumes cell-level bounding boxes in a way that all cells that (a) have the same SR indices having same $y1$ coordinates, (b) have the same SC indices having same $x1$ coordinates, (c) have the same ER indices having same $y2$ coordinates, and (d) have the same EC indices having same $x2$ coordinates. This assumption is necessary for our alignment loss function to work properly. However, different datasets for physical table structure recognition have ground truth annotations defined in different ways. UNLV and ICDAR-2019 archival datasets have ground truth annotated at the cell-level. SciTSR [24] and ICDAR-2013 [48] datasets have ground truth annotation defined at the content-level (cells' bounding box is the smallest rectangle that covers entire content of the cell). To be able to use those for training, we pre-process the ground-truth to obtain cell-level bounding boxes (as explained in Section 3.1). Please

note that this pre-processing step is only done for the training process. Similarly, ground-truth bounding boxes of the synthetic dataset proposed in [12] are provided at the word-level. To obtain cell-level bounding boxes, we use the ground-truth cell adjacency matrix and word-level bounding boxes to obtain content-level bounding boxes. During the testing time in S-A, however, to compute if a detected cell is a true positive, we use the original ground-truth bounding boxes (either at cell-level or content-level), and not the pre-processed ones. Similarly while testing in S-B, we use the original content-level or cell-level bounding boxes as the additional input instead of the pre-processed ones. This ensures consistency while comparing our methods against previously published ones.

Inconsistency with respect to representation of table structure: We broadly classify table structure methods into two categories - (a) physical structure predicting methods that predict cell-level bounding boxes along with their associations [18], [46], [132] and (b) logical structure predicting methods [12], [13], [24], [131] that predict only cell associations. In our work, we standardize our representation as described in Section 3.5, which allows us to directly compare methods in both the experimental setups. To compare the results of TabStruct-Net on logical structure prediction, we generate the mark-up string from the post-processed XML output of TabStruct-Net in the same format as TableBank [13] and PubTabNet [3] by extracting only the structure without cells' coordinates and content.

Inconsistency with respect to evaluation methods: While most of the previously published methods for table structure recognition use precision, recall and F1 scores as described in [48], there are some inconsistencies in this aspect as well. In [12], authors use true positive rate (TPR), false positive rate (FPR) and absolute accuracy on the predicted adjacency matrix to compute performance. In order to standardize evaluation with [12], we infer neighboring cell relations from their output to ensure consistency. Further, [131] use BLEU scores to compare their output with the ground truth. Since our method generates and XML output from the model's predictions, we bring our output to the same format as [131] to ensure a direct and fair comparison on the TableBank dataset [13].

Inconsistency with respect to way of computing evaluation scores: To fairly compare TabStruct-Net against previous methods, we list both micro averaged results on the test datasets. However, it is important to note that for TableBank [13] and PubTabNet [3] datasets, where we evaluate our results on the markup output of the model, we only consider document-averaged results.

3.5 Results on Table Structure Recognition

Tables 3.2 and 3.3 summarize the performance of our model on standard datasets used in the space of table structure recognition.

3.5.1 Analysis of Results

Table 3.4 presents results on ICDAR-2013 dataset. In S-A, we observe that our model outperforms DeepDeSRT [46] method by a 27.5% F1 score. This is because cell coordinates for the latter are obtained by row and column intersections, making it unable to recognize cells that span multiple rows/columns. For dense tables (small inter-row spacing), row segmentation results of DeepDeSRT combined multiple rows into one in several instances. Split+Heuristic [18] method outperforms TabStruct-Net by a small margin, however, it requires ICDAR-2013 dataset-specific cell merging heuristics and is trained on a considerably larger set of images. Therefore, a direct comparison of (Split+Heuristic) with our method is not fair. Nevertheless, comparable results of TabStruct-Net indicates its robustness to ICDAR-2013 dataset, without using any kind of dataset-specific post-processing. However, if compared under the same training environment and no post-processing, our model outperforms SPLERGE with a 3% average F1 score. SPLERGE works well for datasets where ground truth bounding boxes are annotated at the content-level instead of cell-level. This is because it allows for a wider area for a prospective prediction of a row/column separator. Further, since it is based on cell detection through the row and column separators, it is not agnostic to input image noise such as skew and rotations. This method is susceptible to dataset-specific post-processing as opposed to ours, where no post-processing is needed.

Method	**Training**		**Experimental**	**P↑**	**R↑**	**F1↑**
	Dataset	**#Images**	**Setup**			
DeepDeSRT [46]	SciTSR	12K	S-A	0.631	0.619	0.625
SPLERGE [18]	SciTSR	12K	S-A	0.883	0.875	0.879
Split+Heuristic [18]	Private [18]	83K	S-A	**0.938**	**0.922**	**0.930**
TabStruct-Net (our)	SciTSR	12K	S-A	0.915	0.897	0.906
TableNet [131]	Marmot Extended	1K	S-B	0.922	0.899	0.910
GraphTSR [24]	SciTSR	12K	S-B	0.885	0.860	0.872
Split-PDF [18]	Private [18]	83K	S-B	0.920	0.913	0.916
Split-PDF +Heuristic [18]	Private [18]	83K	S-B	0.959	0.946	0.953
DGCNN [12]	SciTSR	12K	S-B	0.972	0.983	0.977
TabStruct-Net (our)	SciTSR	12K	S-B	**0.976**	**0.985**	**0.981**

Table 3.4: Comparison of results for physical structure recognition on ICDAR-2013 dataset. **#Images:** indicates number of table images in the training set. **Heuristic:** indicates dataset specific cell merging rules for various models in [18].

In S-B, TabStruct-Net sets up a state-of-the-art benchmark on the ICDAR-2013 dataset, outperforming all the existing methods [12], [18], [24], [131]. It is further interesting to note that our technique outperforms Split-PDF+Heuristic model also without needing any post-processing. It is because our enhancements to the DGCNN [12] model can capture the visual characteristics of a cell across a larger span through LSTMs. We observe that our model achieves significantly improved performance when content-level bounding boxes are used instead of cell-level, which are much easier to obtain with the

help of OCR tools and PDF meta-information.

CD Network	SR Network	IoU TH	CD Scores			SR Scores		
			P↑	R↑	F1↑	P↑	R↑	F1↑
Mask R-CNN+TD+BU+AL	DGCNN+P2+LSTM	0.5	**0.935**	**0.942**	**0.938**	**0.927**	**0.911**	**0.919**
		0.6	0.921	0.926	0.923	0.915	0.897	0.906
		0.7	0.815	0.820	0.817	0.797	0.785	0.791
		0.8	0.638	0.653	0.645	0.629	0.615	0.622
		0.9	0.275	0.312	0.292	0.247	0.236	0.241

Table 3.5: Physical structure recognition results on ICDAR-2013 dataset for varying IoU thresholds to demonstrate TabStruct-Net's robustness. **ES:** Experimental Setup, **CD:** Cell Detection, **TH:** IoU threshold value, **SR:** Structure Recognition, **P2:** using visual features from P2 layer of the FPN instead of using separate convolution blocks, LSTM**:** use of LSTMs to model visual features along center-horizontal and center-vertical lines for every cell, TD**+**BU**:** use of Top-Down and Bottom-Up pathways in the FPN, **AL:** addition of alignment loss as a regularizer to TabStruct-Net.

Table 3.5 shows the performance of our technique under the varying IoU thresholds. It can be inferred from the table that our model achieves an F1 score of 79.1% on structure recognition with an IoU threshold value of as high as 0.7. For the IoU values of 0.5 and 0.6, our model's performance is 91.9% and 90.6%, respectively. It demonstrates the robustness of our model. Figures 3.7 and 3.8 display some qualitative outputs of our method on the datasets discussed in Section 3.4.1. Figure 3.9 shows some of the failure cases of cell detection by our method. It can be seen that our model fails for table images that have large amounts of empty spaces.

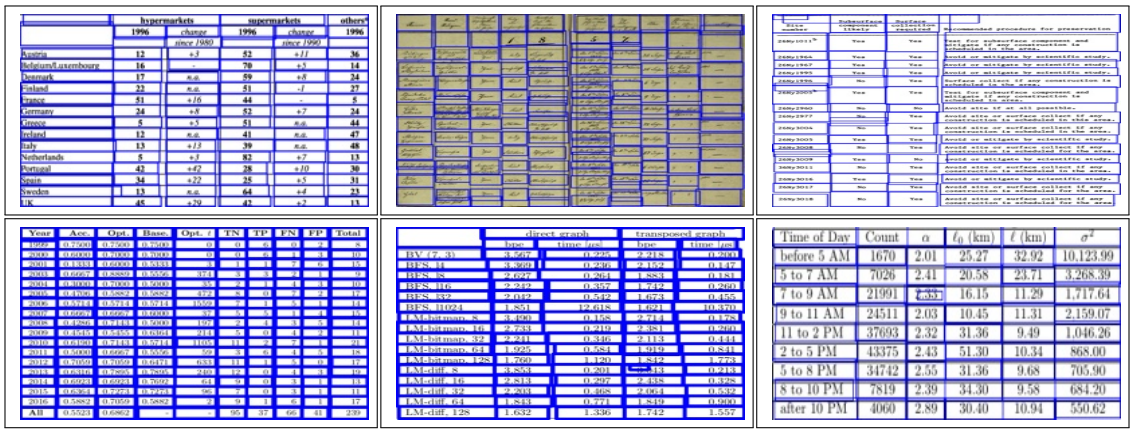


Figure 3.7: Sample intermediate cell detection results of TabStruct-Net on table images of ICDAR-2013, ICDAR-2019 cTDaR and UNLV, SciTSR, SciTSR-COMP and TableBank datasets.

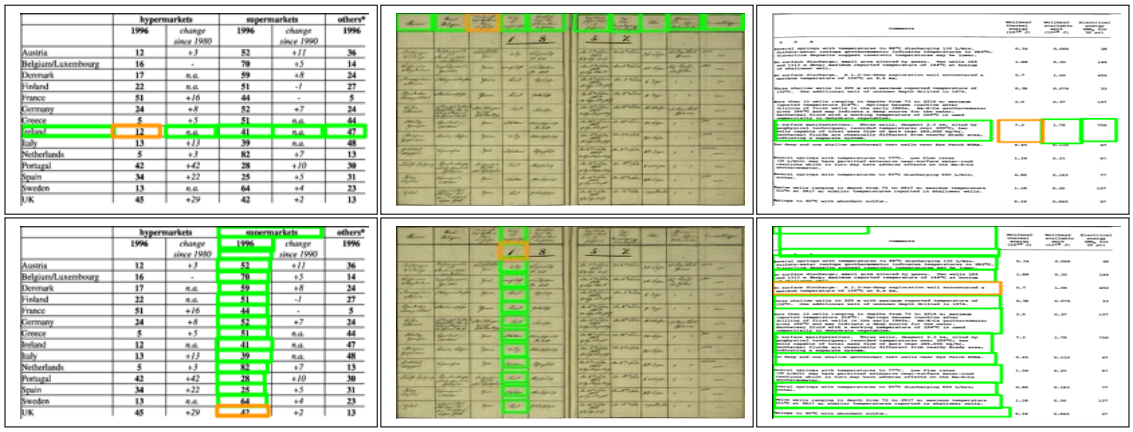


Figure 3.8: Sample structure recognition output of TabStruct-Net on table images of ICDAR-2013, ICDAR-2019 cTDaR archival and UNLV datasets. **First Row:** prediction of cells which belong to the same row. **Second Row:** prediction of cells which belong to the same column. Cells marked with orange colour represent the examine cells and cells marked with green colour represent those which belong to the same row/column of the examined cell.

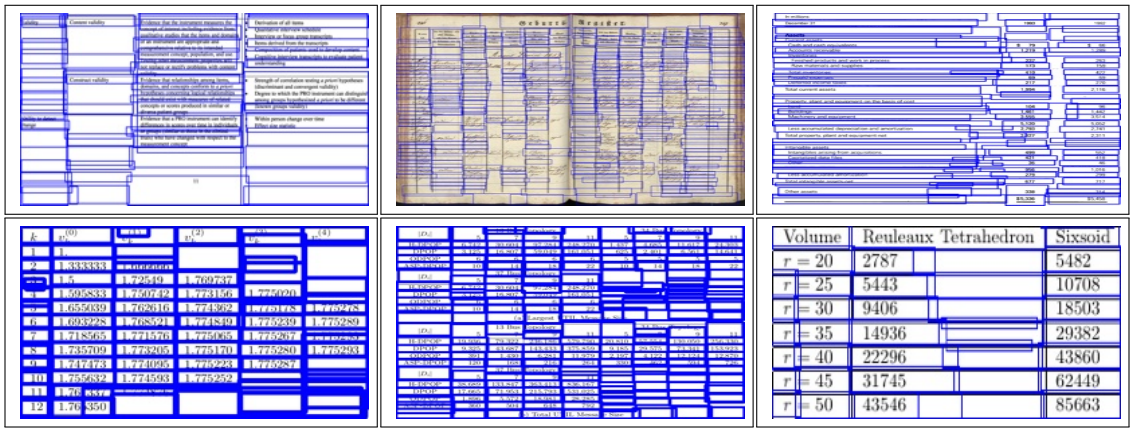


Figure 3.9: Sample intermediate cell detection results of TabStruct-Net on table images of ICDAR-2013, ICDAR-2019 cTDaR, UNLV, SciTSR, SciTSR-COMP and TableBank datasets illustrate failure of TabStruct-Net.

3.5.2 Ablation Study

Table 3.6 shows the outcome of our enhancements to Mask R-CNN [29] and DGCNN [12] models for both cell detection and structure recognition networks under S-A and S-B. From the table, it can be observed that our additions to the networks result in a significant increase of 4% average F1 scores on cell detection and structure recognition tasks. The novel alignment loss, along with the use of top-down and bottom-up pathways in the FPN results in an improvement of 2.3% F1 score for cell detection and 2.4% on structure recognition. Use of LSTMs and P2 layer output to prepare visual features for structure

recognition results in a 2.1% improvement of F1 scores. Interestingly, because both models are trained together in an end-to-end fashion, cell detection's effect is also observed in the form of a 1.5% average F1 score. This empirically bolsters our claim of using an end-to-end architecture for cell detection and, in turn, structure recognition.

ES	CD Network	SR Network	CD Scores			SR Scores		
			P↑	R↑	F1↑	P↑	R↑	F1↑
S-A	Mask R-CNN	DGCNN	0.885	0.890	0.887	0.871	0.860	0.865
	Mask R-CNN	DGCNN+P2	0.886	0.892	0.889	0.877	0.863	0.870
	Mask R-CNN	DGCNN+P2+LSTM	0.898	0.904	0.901	0.885	0.879	0.882
	Mask R-CNN+TD+BU	DGCNN	0.895	0.899	0.897	0.883	0.867	0.875
	Mask R-CNN+TD+BU	DGCNN+P2	0.895	0.901	0.898	0.886	0.870	0.878
	Mask R-CNN+TD+BU	DGCNN+P2+LSTM	0.904	0.910	0.907	0.892	0.884	0.888
	Mask R-CNN+TD+BU+AL	DGCNN	0.905	0.911	0.908	0.891	0.879	0.885
	Mask R-CNN+TD+BU+AL	DGCNN+P2	0.914	0.920	0.917	0.906	0.885	0.895
	Mask R-CNN+TD+BU+AL	DGCNN+P2+LSTM	**0.921**	**0.926**	**0.924**	**0.915**	**0.897**	**0.906**
S-B	-NA-	DGCNN	-NA-	-NA-	-NA-	0.972	0.983	0.977
	-NA-	DGCNN+P2	-NA-	-NA-	-NA-	0.973	0.983	0.978
	-NA-	DGCNN+P2+LSTM	-NA-	-NA-	-NA-	**0.976**	**0.985**	**0.981**

Table 3.6: Ablation study for physical structure recognition on ICDAR-2013 dataset. **ES:** Experimental Setup, **CD:** Cell Detection, **SR:** Structure Recognition, **P2:** using visual features from P2 layer of the FPN instead of using separate convolution blocks, LSTM**:** use of LSTMs to model visual features along center-horizontal and center-vertical lines for every cell, TD**+**BU**:** use of Top-Down and Bottom-Up pathways in the FPN, **AL:** addition of alignment loss as a regularizer to TabStruct-Net.

3.6 Analytical Discussion and Insights

Effect of Alignment Regularization: The alignment term reduces coordinate variance across same-row or same-column cells. Empirically, the standard deviation of boundary positions within each row decreases by 35–40 % compared with a vanilla Mask R-CNN, corresponding to a visible improvement in visual coherence.

Coupling between Detection and Structure: TabStruct-Net's differentiable coupling between the detector and the GNN produces an implicit feedback loop: misaligned detections perturb adjacency predictions, whose gradients then correct the detector. This resembles multi-task co-training, where both tasks act as regularizers for one another. Ablation experiments demonstrate that decoupled training (i.e., freezing detection before GNN training) leads to a drop of 6–8 F1 points on logical-structure metrics.

Comparison with Prior Paradigms: Compared to purely segmentation-based systems such as TableNet or DeepDeSRT, TabStruct-Net avoids over- or under-segmentation by reasoning at the cell-level granularity. Compared to rule-based graph merging (e.g., Split-and-Merge), the learned adjacency classifier

generalizes to unseen layouts. Furthermore, unlike GraphTSR [24], which models adjacency on pre-extracted word boxes, TabStruct-Net remains fully visual, enabling OCR-free structure recovery.

Limitations and Motivation for Extension: Despite its strengths, TabStruct-Net occasionally fails on borderless tables containing large empty regions or skewed alignments. These cases expose limitations of pairwise alignment regularization: without explicit continuity constraints, parallel lines inferred from distant cells may drift. These observations inspired the development of **TabStruct-Net V2** which extend the formulation with continuity, overlap, and rectilinear-adjacency reasoning along with an anchor-free base detector—topics addressed in the subsequent chapter.

3.7 Beyond TabStruct-Net: TOD-Net and TSR-Net

TabStruct-Net demonstrated that a fully visual, end-to-end paradigm can recover table structure without relying on OCR or brittle rule-based post-processing. At the same time, large, real-world document collections surface difficult corner cases: borderless tables with non-uniform spacing, skew and local distortions, multi-line content that expands or contracts cells, and complex row/column spans that upset simple geometric heuristics. In such settings, two recurring pain points emerged.

First, cell localization occasionally produced slightly overlapping or discontinuous cells in dense regions, which created compounding errors for structure recovery. Second, when alignment cues were weak (e.g., faint ruling lines, uneven inter-cell gaps), reasoning purely with local evidence sometimes failed to recover consistent long-range row/column associations.

To address these issues while preserving the strengths of the visual-first approach, we introduced two intermediate modules in our WACV 2022 work: a *Table Object Detection Network* (TOD-Net) that focuses on robust, precise cell localization, and a *Table Structure Recognition Network* (TSR-Net) as shown in Figure 3.10 that converts detected cells into a coherent grid by learning rectilinear row/column relationships. Decoupling localization from structural reasoning let us attack each challenge with targeted inductive biases and training signals. The resulting pipeline substantially reduced downstream error propagation and provided the design substrate for the unified model presented next in *TabStruct-Net V2*.

Design principles: Three principles guided the intermediate design. (i) **Task decoupling:** treat cell localization and structure assembly as distinct prediction problems so that each stage can use the most effective architectures and losses. (ii) **Structural inductive biases:** encode table-specific priors explicitly—rectilinearity, non-overlap of cells, and row/column continuity—so that learning is steered away from implausible configurations. (iii) **Learned, not scripted, assembly:** replace fragile rule cas-

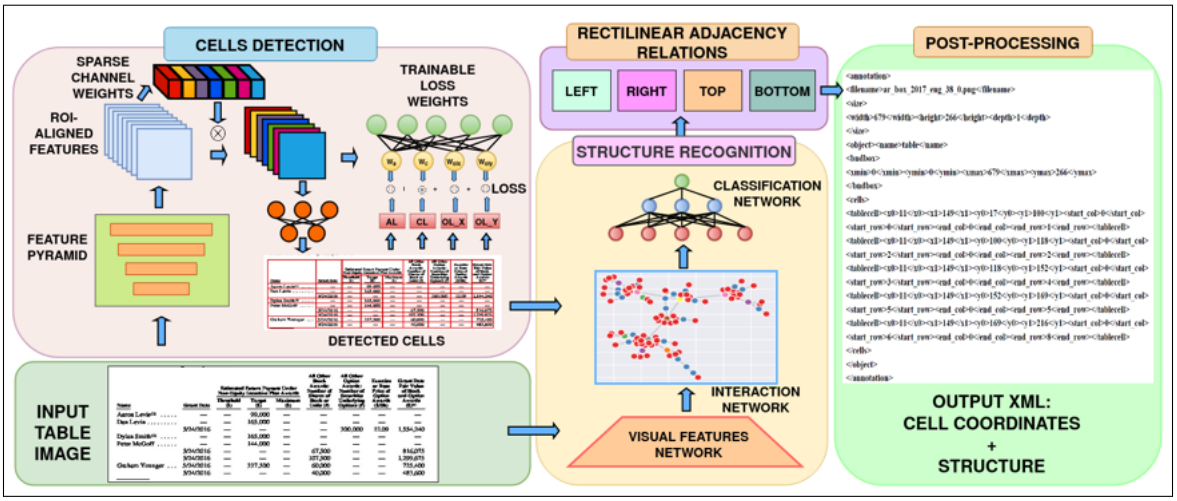


Figure 3.10: Shows our approach. Cell detection is done using TOD-Net. Bounding boxes used as an input by the structure recognition model (based on DGCNN [12], which predicts rectilinear adjacencies. These are then collectively used by the post-processing step to generate output XML containing structure).

cades with a learnable graph of relationships among detected cells, eliminating heavy reliance on line detection or OCR token layout.

TOD-Net: robust and precise cell localization: TOD-Net is a fully convolutional detector specialized for table cells. It extracts multi-scale visual features and augments them with lightweight attention to better aggregate discriminative cues across a cell's interior and boundaries. Two training regularizers proved especially important in difficult layouts: a *continuity* prior that encourages a cell's visual support to be contiguous (discouraging "broken" cells in low-contrast regions) and an *overlap* prior that penalizes competing detections covering the same content (reducing duplicate or partially redundant boxes). These regularizers complement alignment cues and guide the detector toward a tiling of the table region by non-overlapping, gap-free cells.

Compared to the original *TabStruct-Net*, TOD-Net advances three fronts. First, it sharpens localization at borders, particularly where ruling lines are faint or absent; the attention and continuity signals reduce jitter and small holes that previously triggered spurious splits/merges. Second, it decreases *error amplification*: when the detection stage yields a clean, non-overlapping set of cells, structure inference has far fewer degrees of freedom, which stabilizes the final grid under perturbations (e.g., mild skew). Third, by specializing for cell geometry at multiple scales, TOD-Net better handles tables with heterogeneous cell sizes and aspect ratios (e.g., wide header cells over narrow data columns), a longstanding source of fragmentation errors.

TSR-Net: rectilinear, learned structure assembly: Given a set of cell boxes, TSR-Net predicts the latent grid that best explains the layout. Instead of line detection, TSR-Net learns a *rectilinear adjacency* representation: for pairs of cells that are roughly horizontally or vertically aligned (after accounting for small geometric tolerance), it predicts whether they are adjacent along rows and along columns. These pairwise decisions populate a sparse graph over cells. Global consistency constraints and simple, principled heuristics then convert the adjacency graph into row and column groups, infer row-/column-spans, and finally produce a well-formed table grid. Because adjacency is learned directly from pixels and geometry, TSR-Net remains effective on borderless tables and robust to stray marks or imperfect scanning artifacts that often derail rule-based approaches.

Relative to the original *TabStruct-Net*, TSR-Net mitigates two core difficulties. First, it provides **long-range structure** through explicit grouping: by building rows and columns as connected components in learned adjacency graphs, it captures distant associations even when local alignment cues are weak or locally ambiguous. Second, it **reduces post-processing complexity**: rather than layering ad hoc rules after segmentation, TSR-Net's graph-to-grid conversion uses a small set of consistent rectilinear constraints and span-inference steps, which improves reliability and interpretability.

How TOD-Net and TSR-Net address *TabStruct-Net* pain points: The separation of concerns delivers concrete benefits:

- **Cleaner inputs for structure recovery.** With overlap suppressed and continuity promoted, TOD-Net's outputs approximate a tiling of the table region. TSR-Net can therefore operate on a near-canonical set of cells, reducing the need for complex repair logic (e.g., last-minute merges/splits).
- **Stronger global reasoning.** Learned row/column adjacency gives TSR-Net a direct handle on long-range relations, a regime where purely local evidence in the original pipeline sometimes failed.
- **Robustness to borderless and noisy scans.** By steered training (continuity/overlap priors) and line-agnostic assembly (adjacency), the two-stage design is less sensitive to missing rules and minor geometric distortions.
- **Modularity and diagnosability.** Each stage can be evaluated and ablated independently. Failure cases (e.g., a missing cell vs. a mistaken row grouping) are easier to isolate, which informed the architectural choices we integrate in *TabStruct-Net V2*.

What we learned for *TabStruct-Net V2*: The two-stage pipeline revealed an important trade-off: while specialization simplifies learning and improves robustness, the hard boundary between detection and structure still leaves performance on the table. In particular, some structural cues (e.g., that two adjacent cells *should* align into the same row) could have corrected borderline detections *if* gradients

were allowed to flow across stages. Conversely, a slightly mislocalized cell sometimes nudged TSR-Net into an unnecessary split. These observations motivate a *tighter integration* with three themes that directly carry into *TabStruct-Net V2*:

1. **End-to-end structural priors.** The continuity and non-overlap biases that stabilized TOD-Net, and the rectilinear adjacency that powered TSR-Net, should influence predictions *jointly* rather than sequentially. In V2 we embed these biases within a unified objective so that detection and structure reasoning can inform each other during learning (details deferred to the next chapter).

2. **Differentiable structure reasoning.** Instead of treating graph assembly as a post-hoc step, V2 incorporates differentiable surrogates for adjacency and grouping so that the model can "feel" structural penalties during training without resorting to brittle rules.

3. **Global context with local precision.** Attention mechanisms that helped TOD-Net aggregate within-cell context, together with TSR-Net's global grouping signals, can be fused to produce features that are simultaneously sensitive to fine-grained borders and to table-level regularities. This fusion underlies the architectural choices we present in V2.

Conclusion: TOD-Net and TSR-Net were introduced as a purposeful intermediate step: they fix the most consequential failure modes of the original *TabStruct-Net* by (i) specializing cell localization with continuity and overlap-aware learning, and (ii) replacing fragile line- or OCR-driven structure assembly with learned rectilinear adjacency. The architecture in Figure 3.10 captures this separation of concerns and the clean interface between stages. The next chapter takes the final step: it unifies these ideas into a single, end-to-end trainable system that inherits the robustness of the two-stage pipeline while further improving accuracy by sharing features, objectives, and gradients across detection and structure reasoning.

We will refer to the detection-focused ideas (multi-scale features, continuity/non-overlap priors) and the structure-focused ideas (learned rectilinear adjacency, span inference) repeatedly in the next chapter. To keep the exposition cohesive, we intentionally avoid reproducing the loss equations here; they will be introduced and analyzed in the next chapter. Conceptually, V2 can be thought of as a "closing the loop": the strengths of TOD-Net and TSR-Net are preserved, but the model is trained so that structural plausibility and localization precision co-evolve rather than being optimized in isolation.

3.8 Summary

We formulate the problem of table structure recognition as a combination of cell detection (top-down) and structure recognition (bottom-up) tasks. For cell detection, we make a modification to the RPN of original Mask R-CNN and introduce a novel alignment loss function (formulated for every pair of table cells) to enforce structural constraints. For structure recognition, we improve input representation

for the DGCNN network by using LSTM, pre-trained ResNet-101 backbone and RPN of cell detection network. Further, we propose an end-to-end trainable architecture to collectively predict cell bounding boxes along with their row and column adjacency matrices to predict structure. We demonstrate our results on multiple public datasets on both digital scanned as well as archival handwritten table images. We observe that our approach fails to handle tables containing a large number of empty cells along both horizontal and vertical directions. In conclusion, we encourage further research in this direction.

Chapter 4

TabStruct-Net V2: Anchor-Free and Explainable Table Structure Recognition

4.1 Introduction

Table structures in documents are complex yet crucial sources of information. Accurately parsing tables from images enhances the semantic understanding of documents and enables downstream tasks in numerous domains. Although research in table parsing has steadily progressed over the years [1]–[3], [13], [24], [131], [134], the problem remains challenging due to the wide variety of table layouts and content. This surge of interest is driven by the abundance of paper-based documents and the potential of deep learning to solve perceptual tasks that humans perform with ease. Our work in this chapter focuses on *syntactic* table structure recognition, building on our previous TabStruct-Net, TOD-Net and TSR-Net methods [1], [2].

Table structure recognition from images involves identifying all the cell elements in a table and reconstructing their layout and relationships (e.g., which cells are neighbors, which span multiple rows/columns, etc.). This task poses numerous challenges, including multi-row and multi-column spanning cells, multi-line text within cells, empty cells, varying text indentation, absence of ruling lines, and widely varying table densities (from sparse to very dense tables), as illustrated in Figure 4.1. These factors make it difficult to determine cell boundaries and their logical associations.

Our goal is to develop a comprehensive end-to-end solution for **physical table structure recognition** that can robustly handle diverse table layouts. The proposed approach, **TabStruct-Net V2**, is designed to accommodate both dense and sparse tables with high accuracy and efficiency. At a high level, TabStruct-Net V2 follows a two-step reasoning process, inspired by how humans parse tables: a *Top-Down* step that decomposes the table image into individual cells (detection), and a *Bottom-Up* step that associates detected cells to infer the complete table structure (establishing each cell's row and column spans). These two stages are implemented in a single end-to-end trainable model, with explicit constraints that capture intuitive layout rules (such as alignment and continuity of cells) to guide the

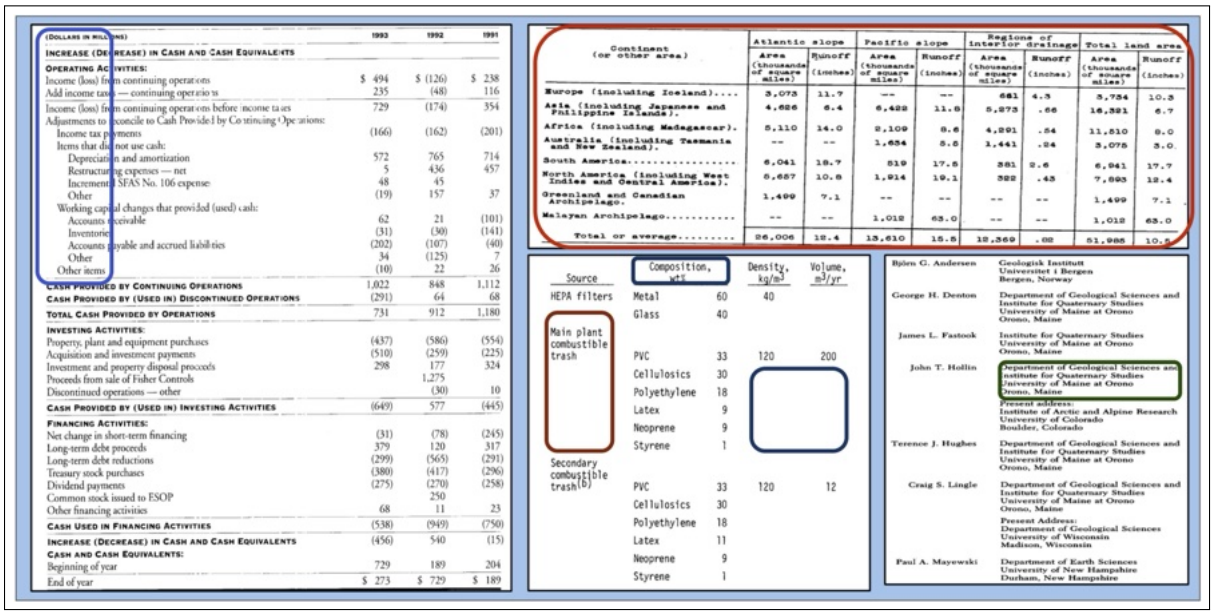


Figure 4.1: Demonstrates the challenges in table structure recognition tasks, including multi-row/column spanning, multi-line, and empty cells.

network's predictions.

In developing TabStruct-Net V2, we draw motivation from human cognitive strategies for reading tables. Humans naturally expect cells in the same row to be aligned horizontally and cells in the same column to be aligned vertically; we also understand that tables should have continuity (no gaps) along rows and columns, and that cells should not overlap each other. We incorporate these notions into the model via additional loss terms that regularize the geometry of predicted cells. Furthermore, unlike approaches that rely heavily on detected text tokens and OCR, our method operates mostly in the vision domain, which helps avoid propagation of OCR errors (such as mis-detected cell boundaries or spurious token splits due to multi-line text) into the structure reconstruction stage.

In TabStruct-Net V2, an input table image is first processed by a Vision Transformer backbone with hierarchical local attention (HLViT) to extract rich visual features at multiple scales. A Top-Down module, built on Deformable DETR, uses these features to detect all table cells as bounding boxes. Importantly, we augment the detection branch with novel **structural constraints**: an *alignment loss* to encourage cells in the same row/column to share boundaries, a *continuity loss* to encourage adjacent rows/columns to align without gaps, and an *overlap loss* to penalize overlapping cell predictions. Simultaneously, a Bottom-Up module predicts the table's structure by computing adjacency relationships between cells: specifically, it predicts four adjacency matrices (left, right, top, bottom) indicating the spatial neighbors for each pair of detected cells. These adjacency matrices encode which cells lie im-

mediately to the left/right or above/below each other in the table grid. Finally, a post-processing step uses the predicted cell bounding boxes and adjacency relations to assemble the complete table structure and output it in a structured format (such as an HTML or XML representation of the table).

In summary, TabStruct-Net V2 is a single, unified network that jointly performs cell detection and structure recognition in an end-to-end manner. The design and training of TabStruct-Net V2 include several technical contributions and improvements over prior work (including our own TabStruct-Net V1).

Technical Contributions: Overall contributions of this work can be summarized as follows:

- **End-to-End, Anchor-Free Architecture:** We develop a single-stage, anchor-free end-to-end trainable architecture that jointly performs cell detection (Top-Down parsing) and table structure recognition (Bottom-Up parsing) within one network. By integrating both tasks, our model can implicitly share information between detecting cells and understanding their relationships, enabling more accurate full table reconstruction.

- **Loss Functions Inspired by Human Perception:** Drawing inspiration from how humans perceive table layouts, we introduce additional regularization losses into the cell detection branch to enforce intuitive structural constraints. In particular, an **alignment loss** encourages cells that belong to the same row or the same column to align perfectly along the horizontal or vertical direction; a **continuity loss** ensures that consecutive cells along a row or column abut each other with no gaps (maintaining table continuity); and an **overlap loss** penalizes any overlapping predictions to maintain the integrity of distinct cells. These losses act as soft constraints that guide the model to predict well-aligned, contiguous, and non-overlapping cell bounding boxes, ultimately improving structure recognition performance.

- **Optimization with Trainable Loss Weights:** We formulate the overall training objective using a min-max optimization strategy with trainable weights for the new structural losses. This allows the model to *learn* how much emphasis to place on each constraint in different regions of an image. The loss weights are dynamically adjusted during training for different regions of interest (e.g., dense areas vs. sparse areas of a table), enabling the model to apply appropriate penalties based on local visual characteristics. This trainable weighting yields a more balanced optimization, improving convergence and performance across heterogeneous table layouts.

- **Hierarchical Local-Attention Transformer Backbone (HLViT):** We design a vision backbone called the **Hierarchical Local-Attention Vision Transformer (HLViT)** that is tailored for high-resolution table images. HLViT applies self-attention in localized windows to capture fine-grained spatial relationships among neighboring cells, and progressively builds a multi-scale feature pyramid through a hierarchy of local attention blocks and down-sampling operations. By focusing attention locally (instead of global attention across the entire image), HLViT significantly improves computational efficiency and scalability to high-resolution inputs (e.g., 800×800 pixels) while

preserving crucial layout details. The hierarchical feature extraction (with four levels of resolution) ensures that both small details (like lines or small gaps) and larger structures are represented in the features. This backbone effectively combines the strengths of convolutional inductive bias (locality) with the representational power of transformers.

- **Faster Adjacency Prediction with Self-Attention:** In the Bottom-Up structure recognition stage, we replace the graph neural network used in TabStruct-Net V1 with a simpler and faster self-attention mechanism to predict cell adjacency. The original TabStruct-Net employed a dynamic graph convolution network (DGCNN) [12] that required sampling neighbors and iteratively updating embeddings, which became a bottleneck. In TabStruct-Net V2, we instead use a single-head self-attention layer (without softmax) operating on the cell features to directly compute adjacency scores between every pair of cells. This change eliminates the expensive neighbor-sampling procedure and reduces the complexity of adjacency prediction to basic matrix operations, resulting in a substantial speedup during inference (and training) while maintaining accuracy in capturing cell relationships.

- **Heuristic-Driven Data Augmentation:** We propose a novel data augmentation technique to address the scarcity of annotated complex tables and to improve cross-domain robustness. Noticing that scientific tables often have similar header structures but fewer rows compared to business tables (which are dense with numeric data), we augment training data by vertically concatenating certain tables with themselves two or three times (keeping only one header) to create ẗallerẗables. This simulates the visual density of long financial tables while preserving scientific table styles, bridging the gap between different table domains. This augmentation is applied selectively to diversify the training set, making the model more resilient to differences between, e.g., scientific article tables and business document tables.

- **Curated Training Dataset for Visual Diversity:** We develop an automatic **dataset curation algorithm** to assemble a new training dataset of 25,000 table images by combining and filtering existing datasets. The algorithm selects samples to ensure a balanced coverage of various visual characteristics (e.g., a range of table sizes, percentages of empty or spanning cells, etc.). By prioritizing complex tables that are under-represented in individual datasets, this curated dataset enables training a single model that generalizes better across domains. In our experiments, we find that models trained on this curated data perform consistently better, especially on unusual or challenging table layouts.

- **Robust Evaluation Protocol:** We design a comprehensive evaluation protocol to assess the robustness of table structure recognition models under varying conditions. We categorize test tables by distinct challenges (such as percentage of empty cells, number of spanning cells, table size, etc.), and evaluate performance across a spectrum of increasingly difficult conditions. This analysis provides deep insights into the model's strengths and failure modes, and helps guide future

research by highlighting which visual factors most affect performance. We also introduce a split-and-merge inference strategy (described below) to successfully handle extremely large or dense tables that would otherwise exceed memory limits or confuse the model.

- **Split-and-Merge Strategy for Large Tables:** To handle very large or dense tables during inference, we propose an effective **split-and-merge strategy** that does *not* require any change to the trained network architecture. Before feeding a table image into TabStruct-Net V2, we perform a simple preprocessing: we run an off-the-shelf OCR text detector on the image to identify text lines, and if the table is extremely large (e.g., has many rows that cannot fit into the 800×800 input at a readable scale), we split the table image horizontally into smaller segments:contentReference[oaicite:9]index=9. Each segment contains at most 10 lines of the table, and we include a 2-line overlap with adjacent segments (i.e., each split shares a couple of rows with its neighbors) to preserve context for merging. We then apply the model to each split image separately, and finally merge the partial results by using the overlapping regions to align and join the detected cells from each split. This strategy allows us to accurately parse tables that are far larger than the network's normal input size, effectively extending the model's applicability to "unbounded" table sizes without retraining.

The above contributions mark significant improvements over our original TabStruct-Net V1 model. To put these advances in context, we briefly review the TabStruct-Net V1 approach and its limitations. TabStruct-Net V1 [1] introduced the top-down and bottom-up parsing framework for tables, using a ResNet-based backbone and an object detection model (Mask R-CNN) to detect cells, followed by a graph neural network (DGCNN) to infer adjacency relationships. While it demonstrated the viability of end-to-end table structure recognition, TabStruct-Net V1 had several shortcomings: (1) it relied on generic CNN features which struggled with very high-resolution inputs and fine visual details (missing subtle misalignments); (2) it did not explicitly enforce alignment or continuity, so it could produce slightly misaligned cell boxes that led to downstream errors in structure; (3) the use of a sampling-based GNN for adjacencies was computationally expensive and introduced a bottleneck during inference; and (4) training on limited datasets caused bias towards common structures (e.g., tables with few or no spanning cells), reducing robustness on unusual table layouts. In TabStruct-Net V2, we directly address each of these issues via our new HLViT backbone, the alignment/continuity/overlap losses, the self-attention adjacency module, and curated data and augmentation strategies. As a result, TabStruct-Net V2 achieves substantially higher accuracy, especially on complex tables, and is an order-of-magnitude faster in adjacency prediction, as will be evidenced in the Results section. We will detail each component of the TabStruct-Net V2 methodology in the following sections, then describe the experimental setup and present comprehensive evaluations comparing TabStruct-Net V2 to prior art (including TabStruct-Net V1) on multiple benchmark datasets.

4.2 Datasets and Dataset Curation

A robust table structure recognition model should handle the diverse visual characteristics that tables exhibit. However, obtaining large annotated datasets that cover *all* variations is challenging due to high annotation cost. Most existing datasets have biases because they were semi-automatically generated (e.g., by parsing PDF documents) and may not capture certain layout constraints that humans expect. For instance, automatically generated cell annotations often use text token bounding boxes as cell proxies, which can be misaligned or fragmented, and they omit empty cells entirely (since empty cells have no text). Such issues can lead to training a model that, for example, tolerates misaligned cells or fails to predict empty cells. Therefore, careful dataset preparation and augmentation are critical for improving model robustness.

In our work, we leverage two major public datasets that have been widely used for table structure recognition with deep learning: **FinTabNet** [4] and **SciTSR** [24]. FinTabNet (FTN) consists of $\sim$90k tables extracted from financial documents (primarily reports and statements), typically featuring dense numeric tables with relatively regular structures. SciTSR contains $\sim$12k tables from scientific research articles, which often include more varied layouts, complex spanning structures, and are generally less dense than FinTabNet's tables. These two datasets differ significantly: FinTabNet tables are usually larger and densely populated (with many rows/columns and numeric content), whereas SciTSR tables are smaller, text-heavy and may have complex column or row headers. This disparity is evident in the distribution of various visual characteristics in each dataset.

Figure 4.2 shows the distribution of certain key table characteristics in FinTabNet (blue) vs. SciTSR (orange) versus a *curated* dataset we assembled (green). For example, SciTSR has a higher fraction of tables with multi-row/column spanning cells (due to complex scholarly tables), while FinTabNet has more tables with empty cells and generally higher cell counts (dense financial tables). Training a model on one of these datasets alone often leads to uneven performance: it may generalize poorly to the other domain or to edge cases. Indeed, as we will later analyze, a model trained only on SciTSR struggles on very large or numeric tables, whereas one trained on FinTabNet may miss complex spanning layouts.

To address the limitations of individual datasets, we curate a new combined training set of 25,000 tables that balances these visual factors. Our **dataset curation algorithm** (Algorithm 1) selects tables from the union of FinTabNet and SciTSR (and a few others) such that for each of several criteria (table density, fraction of empty cells, fraction of multi-line text cells, fraction of spanning cells, etc.), we include a representative spread of examples. The criteria we consider and the algorithm's procedure are detailed in Algorithm 1. In essence, we bin the tables by each characteristic and then greedily select tables from the highest complexity bins for each criterion, cycling through criteria to ensure diversity. By prioritizing more complex tables (while still including simpler ones), the curated dataset exposes the

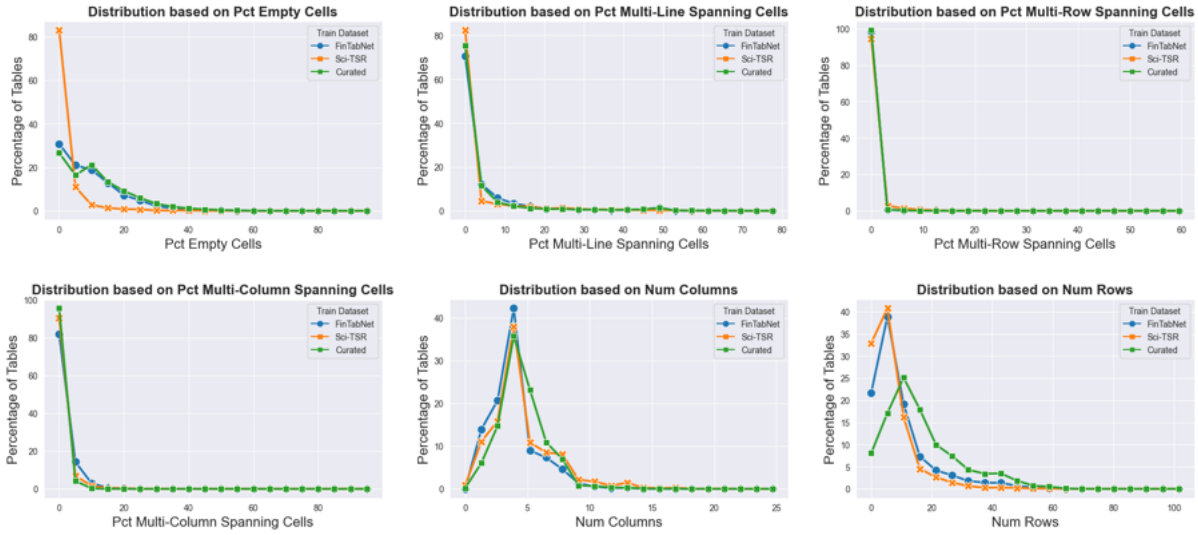


Figure 4.2: Shows the distribution of visual characteristics in three training datasets: FintTabNet (blue), SciTSR (orange), and Curated (green). Each sub-graph represents a specific characteristic, with the X-axis showing its absolute value and the Y-axis the percentage of tables.

model to a wide variety of scenarios, making it more *cross-domain adaptive*.

We analyze the composition of this curated dataset in Figure 4.2. As shown, the curated set (green curves) provides a more uniform coverage across different complexity levels, whereas the original FinTabNet (blue) and SciTSR (orange) were highly skewed for certain attributes (e.g., SciTSR had very few empty cells, FinTabNet had very few large spanning cells). The impact of this balanced training data will be demonstrated in our results, where the model trained on the curated data outperforms those trained on single sources, especially in high-complexity cases.

Our curation strategy prioritizes a holistically challenging dataset over simply maximizing the percentage of any single complex feature. Our prioritized, greedy curation strategy selects the most complex samples based on a specific hierarchy:

1. Table Density (Rows and Columns)
2. Percentage of Empty Cells
3. Percentage of Multi-line Spanning Cells
4. Percentage of Multi-row Spanning Cells
5. Percentage of Multi-column Spanning Cells

Algorithm 1 Training Dataset Curation

1. **Initialization:** Define a set of table complexity criteria (e.g., number of rows, number of columns, % of empty cells, % of multi-line cells, % of multi-row spanning cells, % of multi-column spanning cells for each table). Compute these statistics for every table in the pool of candidate datasets (e.g., FinTabNet + SciTSR).
2. **Statistics Gathering:** For each numerical criterion:
   - Divide the range of non-zero values into k equal-sized bins.
   - Assign each table to the appropriate bin based on its value for this criterion; record counts for each bin.
3. **Dataset Curation:** Sort the criteria in decreasing order of importance (for our purposes, we prioritize overall table density first, then percentages of empty cells, multi-line, multi-row, multi-col spans, etc.). Then:
   - Initialize an empty curated dataset.
   - **While** the curated dataset size is less than the desired target:
     - For each criterion in priority order:
       * Identify the highest non-empty bin for that criterion (i.e., the most complex remaining tables for this aspect).
       * Add a batch of tables from this bin to the curated dataset (and mark those tables as used).
       * Decrease the bin index (to include next-most complex tables next).

In addition to FinTabNet and SciTSR, we use several standard benchmarks for evaluation in this chapter. The **ICDAR 2013** table competition dataset (ICDAR-2013) [48] is a smaller set of tables (from EU and UN documents) often used for evaluating table structure recognition; it contains both tabular data and forms, and often requires adaptation of trained models via fine-tuning. We follow the evaluation protocol of prior works [1], [144], [145] for ICDAR-2013 by fine-tuning on a subset and testing on the rest. The **ICDAR 2019 cTDaR** dataset [41] comprises modern and historic tables; we focus on the modern track for structure recognition and evaluate using the standard IoU-based metrics at threshold 0.5. We also report results on the **PubTabNet** dataset [3] (denoted PTN), which is a large-scale dataset of table images from scientific publications with structure ground truth, commonly evaluated using the TEDS metric (described later). Finally, note that the **WTW** dataset [22] (wild tables with arbitrary rotations) is not included in our evaluation, because our method assumes rectilinear table structure (rows roughly horizontal). WTW contains many curved or rotated tables which violate the standard top-down structure assumptions (e.g., cell alignment), so for fairness we exclude it from evaluation.

Figure 4.2 summarizes the training dataset distributions, while Figure 4.3 shows the distribution of key characteristics in the various *test* sets (ICDAR-2013, FinTabNet-Test, SciTSR-Test, etc.). These plots highlight differences such as: SciTSR-Test tables are generally simpler (hence its distribution skews toward lower complexity in each criterion) whereas FinTabNet-Test and ICDAR-2013 include

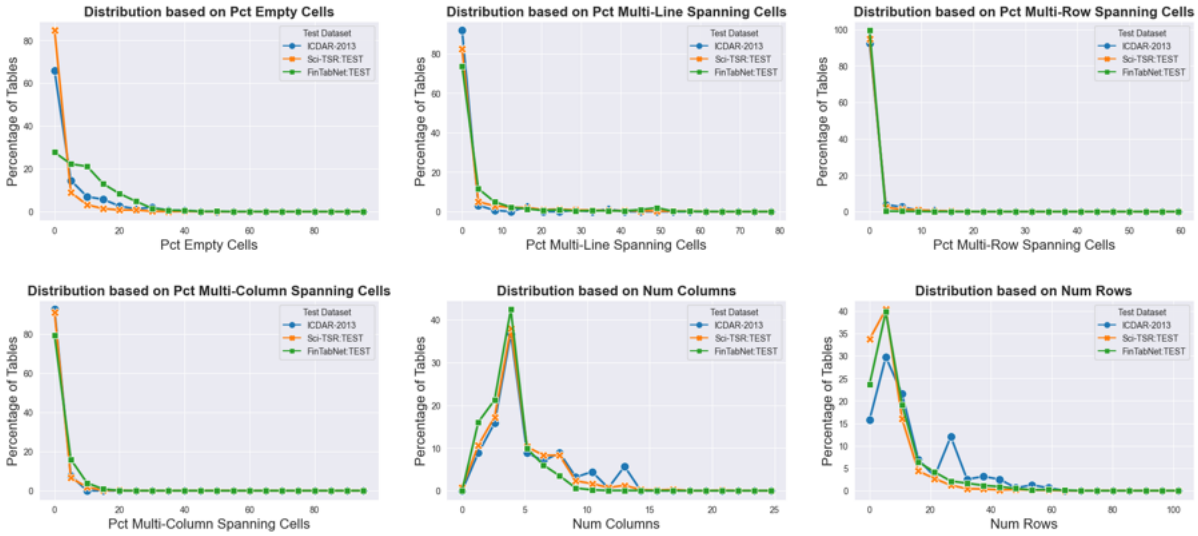


Figure 4.3: Presents the distribution of visual characteristics in three training datasets: ICDAR-2013 (blue), FinTabNet-Test (orange), and SciTSR-Test (green). Each sub-graph represents a specific characteristic, with the X-axis showing its absolute value and the Y-axis the percentage of tables. For a clearer view, zoom in.

more challenging cases. This analysis justifies the need for a broad training set and will later aid in interpreting the results per characteristic.

4.3 TabStruct-Net V2

Given an input image I of a table, the goal of TabStruct-Net V2 is to produce a structured representation of the table that includes every cell's coordinates and its row and column span within the table. We represent each detected cell TC_i (for $i = 0, 1, \ldots, n-1$, where n is the number of cells) by its bounding box $(X_l^i, Y_t^i, X_r^i, Y_b^i)$ corresponding to left, top, right, and bottom boundaries, along with structural indices (SR_i, SC_i, ER_i, EC_i) indicating the cell's start row, start column, end row, and end column in the table grid. The challenge is that cell detection and structure recognition are tightly interdependent: detecting a cell correctly can depend on understanding neighboring cells (to know where one cell ends and another begins), and conversely, determining adjacency or spanning relations requires knowing the cell boundaries. Rather than treating these subproblems in isolation, TabStruct-Net V2 addresses them jointly in a single network so that information is shared between them during training and inference.

Our approach consists of two coordinated parts: a **Top-Down module** that localizes all table cells, and a **Bottom-Up module** that predicts adjacency relationships between those cells. The model architecture is depicted in Figure 4.4. The Top-Down module uses an object detection paradigm (specifically Deformable DETR) to produce a set of cell bounding box predictions. The Bottom-Up module takes features of the detected cells and produces four $n \times n$ adjacency matrices: $\mathbf{M}^L$, $\mathbf{M}^R$,

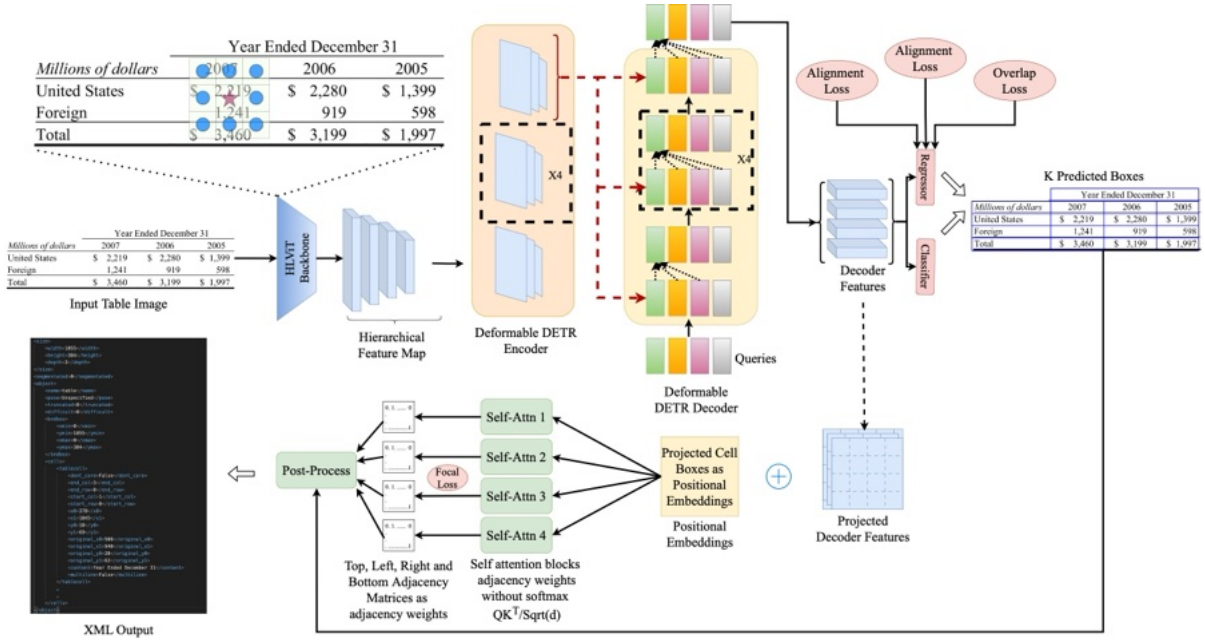


Figure 4.4: Presents architectural diagram of TabStruct-Net V2. The input image is passed through HLViT, a hierarchical local-attention aware vision transformer, output of which is subsequently utilized by Deformable DETR model augmented by table cells specific loss functions. Corresponding to each query, four adjacency matrices are predicted using self attention heads without the softmax function. Finally, the adjacency matrices and the cell bounding boxes are utilized by the postprocessor to predict table structure as an XML.

$\mathbf{M}^T$, $\mathbf{M}^B$, corresponding to left-neighbor, right-neighbor, top-neighbor, and bottom-neighbor relations respectively. For example, if cell j is immediately to the right of cell i in the same row, then $M^R_{i,j} = 1$ (and similarly $M^L_{j,i} = 1$), otherwise 0. These adjacency matrices provide a complete description of the table's grid structure when combined with the cell coordinates. Finally, a post-processing algorithm takes the predicted boxes and adjacency matrices and computes each cell's (SR, SC, ER, EC) span indices, resolving any inconsistencies by simple geometric reasoning.

Formally, the model can be seen as learning a function $f : I \rightarrow (b_i, a_i)i = 0^{n-1}$ where b_i are bounding boxes and a_i are adjacency vectors (the i-th row/column of each adjacency matrix), which are then decoded into the final table structure. The network is trained end-to-end to minimize a composite loss: $\mathcal{L} = \mathcal{L}_{\text{DETR}}\left(\{b_i\}, \{b_i^{gt}\}\right) + \lambda_{\text{adj}}\,\mathcal{L}_{\text{adj}}\left(\{\mathbf{M}^L, \mathbf{M}^R, \mathbf{M}^T, \mathbf{M}^B\}, \{\mathbf{M}^{gt}\}\right) + \sum_{\text{structure}} \lambda_{\text{struc}}\,\mathcal{L}_{\text{struc}}$, where $\mathcal{L}_{\text{DETR}}$ includes standard detection losses (classification and box regression), $\mathcal{L}_{\text{adj}}$ is a loss on adjacency matrix predictions (we use a sigmoid focal loss), and $\mathcal{L}_{\text{struc}}$ are the new structural losses (alignment, continuity, overlap) we introduce, with weights λ_{struc}. We will detail each component next.

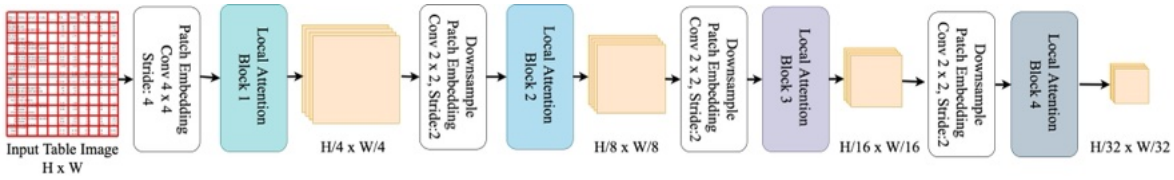


Figure 4.5: Presents backbone architecture of TabStruct-Net V2, which we term as Hierarchical Local-Attention Aware Vision Transformer (HLViT). Each point in the feature map attends to its local neighborhood only. Each block of HLViT downsamples the spatial size of the feature map by a factor of 2. All the four feature maps are utilized by the Deformable DETR.

An important aspect of our detection approach is that we treat it as **anchor-free**: we do not rely on pre-defined anchor box positions or sizes. Instead, the transformer-based detector directly predicts absolute bounding boxes. We also avoid the use of heuristics like morphological post-processing or detecting ruling lines; all structural reasoning is encapsulated in the learning framework via the losses and bottom-up module.

Figure 4.4 shows how the pieces integrate. In the following subsections, we describe: (1) the HLViT backbone for feature extraction; (2) the Top-Down cell detection, including the Deformable DETR model, the stable matching and classification loss, and our structural alignment/continuity/overlap losses; (3) the Bottom-Up structure recognition with adjacency matrix prediction; and (4) the post-processing stage that produces the final structured table output. Key notations and concepts are summarized along the way for clarity.

4.3.1 Backbone: Hierarchical Local-Attention Vision Transformer (HLViT)

The backbone of TabStruct-Net V2 is a specialized Vision Transformer that employs *local self-attention* in a hierarchical feature pyramid. We term this the Hierarchical Local-Attention Vision Transformer (HLViT). The design of HLViT is motivated by the need to efficiently handle high-resolution images (tables can be large and detailed) while preserving locality of interactions (since neighboring cells and text are most relevant to each other).

HLViT, as shown in Figure 4.5 is composed of four sequential **Local Attention Blocks**, each followed by a 2×2 patch embedding (down-sampling) layer. The input image I (e.g., of size 800×800 pixels) is first split into non-overlapping patches and linearly projected to an initial set of feature tokens. These tokens then pass through the local attention blocks. Within each local attention block, self-attention is computed *only among spatially neighboring tokens*. In our implementation, we choose a fixed neighborhood size (each token attends to a 3×3 window of tokens around it, i.e., 8 neighbors plus itself). This significantly reduces the complexity of self-attention (from $O(N^2)$ to $O(N)$ per layer, where N is number of tokens) and focuses the model on local spatial patterns such as line continuity or

small gaps between cells.

Algorithm 2 Neighborhood Extraction with Convolutional Operations

Input: Tensor $x \in \mathbb{R}^{B\times H\times W\times C}$ where B is the batch size, H and W are the height and width of the input, and C is the number of channels

Output: Transformed tensor after applying neighborhood extraction

1: Initialize kernel height k_h, kernel width k_w, number of channels C, stride s, and padding p
2: conv_param $\leftarrow$ Identity$(k_h \times k_w)$
3: conv_param $\in \mathbb{R}^{1\times k_h\times k_w\times (k_h\times k_w)}$
4: conv_param $\in \mathbb{R}^{C\times k_h\times k_w\times (k_h\times k_w)}$
5: conv_param $\in \mathbb{R}^{(C\times k_h\times k_w)\times k_h\times k_w\times 1}$
6: out $\leftarrow$ conv2d(x, conv_param, stride=s, padding=p, groups=C)
7: out $\leftarrow$ out.reshape$(B, H \times W, C, k_h, k_w)$
8: out $\leftarrow$ out.transpose$(0, 1, 3, 4, 2)$
9: **Return:** out

One bottleneck in implementing local (or deformable) attention is efficiently gathering the neighbors for all query positions. Inspired by the Slide-Transformer approach [146], we avoid custom CUDA kernels and instead leverage standard convolution operations to extract local neighborhoods for all tokens in parallel. In particular, we use a group convolution with a specially constructed identity kernel to achieve an *im2col* operation: it rearranges the feature map such that for each token (at each spatial position), we obtain a $k_h \times k_w$ grid of its neighbors (for a chosen local window size $k_h \times k_w$). The steps of this neighborhood extraction are outlined in Algorithm 2. Using this technique (which is effectively a clever use of convolution to collect local patches), we significantly speed up the local attention operations and make them GPU-efficient without writing custom low-level code.

Each Local Attention Block in HLViT, as shown in Figure 4.6 performs self-attention over these extracted local patches. That is, each token attends only to the tokens in its 3×3 vicinity (including itself). This focuses the model on fine-grained spatial context, which is ideal for table structure recognition because many relationships (like whether two cells are adjacent or separated) depend on local neighboring pixels (for example, a small gap vs. a line can differentiate separate cells). After each local attention block, we reduce the spatial resolution by a factor of 2 (in both height and width) via patch embedding (which is effectively a strided convolution or pooling) to build a hierarchical representation. Thus, HLViT outputs four feature maps at $1/2$, $1/4$, $1/8$, and $1/16$ of the input resolution, each with an increasing channel depth. These multi-scale features are analogous to a CNN's feature pyramid, and we feed all of them into the Deformable DETR module so that it can detect cells of various sizes using appropriate feature scales.

It is worth noting that **local attention** in ViTs not only reduces computation but also inherently biases the model to focus on spatially contiguous regions. For tables, this is advantageous because adjacent

Figure 4.6: Presents a Local Attention Block of HLViT. Each point in the feature map attends to its local neighborhood only.

cells have strong interactions (e.g., sharing borders). By contrast, global attention would waste effort modeling long-range dependencies that might be less relevant or could be captured by subsequent graph reasoning. The HLViT architecture effectively mirrors the hierarchical nature of tables: it first captures very local patterns (like cell boundaries, ruling lines) and gradually builds up to coarser, more abstract features (like entire rows or columns). By preserving spatial information through this hierarchy, HLViT lays a strong foundation for accurate cell detection.

Figure 4.5 illustrates the HLViT backbone architecture. Each point in the feature map at a given stage attends only to its local neighborhood (depicted by the small patch around it), and each subsequent stage operates on a down-sampled version of the image. All four resulting feature maps (at different scales) are input to the Deformable DETR so that it can query features at the appropriate scale for each potential cell (Deformable DETR inherently can attend to multi-scale feature maps).

The output of the backbone is therefore a set of multi-scale feature embeddings F^1, F^2, F^3, F^4, where F^1 is high-resolution (e.g., 200×200 if input is 800×800) and F^4 is lowest resolution (50×50 in this example). These features are next used by the Top-Down detection transformer.

4.3.2 Top-Down Cell Detection

For detecting table cells in the image (the Top-Down step), we adopt an **object detection transformer** approach. Specifically, we use the Deformable DETR model [147] as the foundation for our cell detector, due to its ability to efficiently attend to relevant regions of multi-scale feature maps. Deformable DETR (DDeTr) generates a fixed set of learnable *queries* which attend to the feature maps $F^1, \ldots, F^4$ produced by HLViT and predict bounding boxes and class confidences for each query. Since our task is to detect table *cells*, we treat each query as either predicting one cell or a "no object" (null)

prediction. Thus, all queries output either a cell or no-cell; we do not need multiple classes, just a binary label for each query indicating whether it corresponds to a table cell or not. We set the number of DETR queries high enough to cover the maximum expected number of cells in a table (in practice, 400 queries sufficed for our datasets, which typically contain fewer than 200 cells on average).

A key difference from natural scene object detection is that table cells are not independent entities—they form a structured group. Traditional DETR uses a one-to-one matching (via the Hungarian algorithm) between predicted and ground truth boxes, based on classification and box overlap, but it does not incorporate any notion of structure among objects. To inject structural awareness, we incorporate our special loss terms (alignment, continuity, etc.) as described shortly. Another modification we make is in the matching cost: instead of the standard classification score in Hungarian matching, we employ **Stable Hungarian Matching** as proposed in [148]. In stable matching, the classification part of the matching cost is defined using a *position-based metric* (like IoU between prediction and ground truth) rather than the classifier confidence alone. This is beneficial in our case because all cells are of the same class, and we care more about assigning predictions to ground truths based on how well the predicted box overlaps the true cell. We use IoU as the position metric in the matching cost to make the assignment of predictions to ground-truth cells more stable and informed by geometry, which improved convergence (we found our detector converges in 2 epochs with this strategy, whereas original Deformable DETR took 12+ epochs).

The loss for cell detection includes the standard components from Deformable DETR: a **bounding box ℓ_1 loss** and a **generalized IoU (GIoU) loss** [147] for the regression, and a **classification loss** for predicting whether a query is a cell or background. We denote these as $\mathcal{L}bbox$ and $\mathcal{L}GIoU$ and $\mathcal{L}cls$ respectively. We use the same formulations as in [147] for $\mathcal{L}bbox$ and $\mathcal{L}GIoU$. For $\mathcal{L}cls$, with stable matching, we adapt the focal loss from [148]: if a prediction i is matched to a ground truth cell, we want it to predict class=cell with probability close to $f(s_i)$, where s_i is a position-based metric (IoU) for that match. If prediction j is matched to no object (negative), we want its probability p_j of being cell to be low. The classification loss for a batch can be expressed as:

$$L_{\text{cls}};=;\sum_{i=1}^{N_{pos}}\big(|f(s_i)-p_i|^{\gamma}\cdot \text{BCE}(p_i,;f(s_i))\big);+;\sum_{j=1}^{N_{neg}} p_j^{\gamma}\cdot \text{BCE}(p_j,;0),, \tag{4.1}$$

where BCE is binary cross-entropy, N_{pos} and N_{neg} are numbers of matched and unmatched queries, and $f(s)=\epsilon(s^2)$ is a rescaling function (we use the same form as [148] with ϵ being a small constant) to map IoU s to a target probability. γ is the focal loss focusing parameter (we use $\gamma=2$). Intuitively, if a predicted box has high IoU with a ground truth, we encourage its confidence to be high and penalize deviation from that target. Unmatched predictions incur loss proportional to p_j^{γ}, concentrating the loss on those with higher false confidence. This stable classification loss (derived from [148]) replaces the standard cross-entropy on a fixed "object" vs "no-object" target of 1 or 0; it improved training stability

and convergence speed in our experiments.

With these detection losses, TabStruct-Net V2 can already detect most table cells accurately. However, raw detection alone might still produce slight misalignments or overlaps, especially for difficult cases such as tables without grid lines or with spanning cells. To further refine the detection with structural knowledge, we introduce the following regularization losses:

Alignment Loss: This loss, denoted L_{al}, imposes that cells that begin or end on the same row should share the same y-coordinates, and cells that begin or end in the same column should share the same x-coordinates. In other words, if two cells are in the same physical row of the table, their top edges should align (and if they span to the same ending row, their bottom edges align). Similarly for columns. We derive four components for alignment:

$$\begin{aligned}
\mathcal{L}_{SR} &= \forall r \in SR \sum_{i,j \in \, row \; r} ||Y_t^i - Y_t^j||_1^1, \\
\mathcal{L}_{ER} &= \forall r \in ER \sum_{i,j \in \, row \; r} ||Y_b^i - Y_b^j||_1^1, \\
\mathcal{L}_{SC} &= \forall c \in SC \sum_{i,j \in \, col \; c} ||X_l^i - X_l^j||_1^1, \\
\mathcal{L}_{EC} &= \forall c \in EC \sum_{i,j \in \, col \; c} ||X_r^i - X_r^j||_1^1,
\end{aligned}$$

where $|\cdot|$ denotes absolute difference. In practice, we compute these sums over the ground truth row/col groupings and apply them on the predicted coordinates of matched cells. We then define the alignment loss as:

$$\mathcal{L}_{al} = \mathcal{L}_{SR} + \mathcal{L}_{ER} + \mathcal{L}_{SC} + \mathcal{L}_{EC} \tag{4.2}$$

In effect, L_{al} forces the detector to output bounding boxes that line up perfectly in a grid formation for those cells that are supposed to share a boundary. This is especially useful for tables without drawn grid lines: the model learns to infer alignment from content and spacing cues. It also helps in cases of slight skew in predictions, by pulling boxes into alignment.

Continuity Loss: The continuity loss L_{cl} is introduced to ensure that adjacent cells in a row or column meet without any gap (unless separated by a spanning cell, which is handled by the adjacency relations rather than detection). The rationale is that if cell j is immediately to the right of cell i (i.e., j starts in the column immediately after i ends), then the right boundary of i should equal the left boundary of j (i.e., $X_r^i = X_l^j$). Similarly, if one cell is directly below another (adjacent in the next row), the bottom of the top cell should equal the top of the bottom cell ($Y_b^i = Y_t^j$). We encode this as:

$$\begin{aligned}
\mathcal{L}_r &= \sum_{i,j \in \, cells} ||Y_t^i - Y_b^j||_1^1 \cdot \mathbb{I}\big(SR_i == ER_j + 1\big), \\
\mathcal{L}_c &= \sum_{i,j \in \, cells} ||X_l^i - X_r^j||_1^1 \cdot \mathbb{I}\big(SC_i == EC_j + 1\big) \\
\mathcal{L}_{cl} &= \mathcal{L}_r + \mathcal{L}_c,
\end{aligned} \tag{4.3}$$

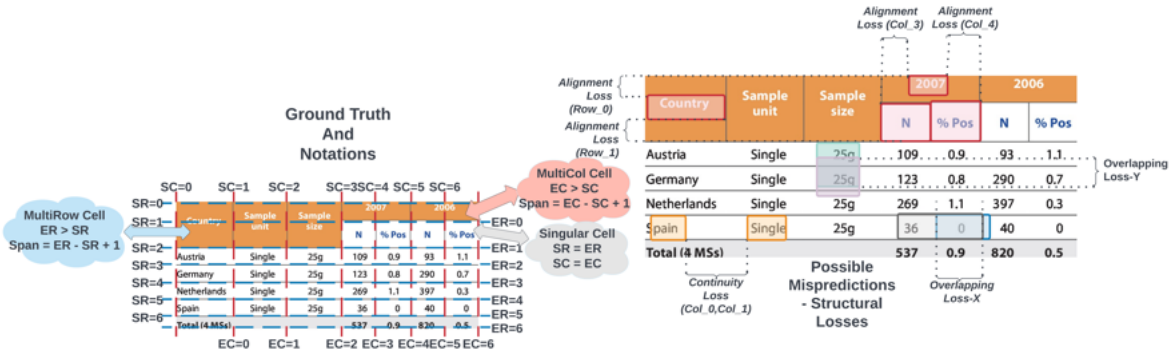


Figure 4.7: Presents the notations of Start Row (SR), End Row (ER), Start Column (SC), and End Column (EC) indices are annotated in the ground truth. Please note that for single row and column cells, SR=ER and SC=EC. For cells span multiple rows, ER > SR, and for those that span multiple columns, EC > SC. The image on the right side demonstrates various types of miss predictions for a cell and how various structural losses (alignment, continuity, overlap in X direction, and Y direction) are computed between every predicted pair of cells.

where $\mathbb{I}[\cdot]$ is an indicator function equal to 1 when its condition is true and 0 otherwise. This encourages each row of cells to form a continuous horizontal sequence with no gaps, and similarly for each column. It is particularly beneficial for tables with irregular spacing or where one cell spanning multiple rows might create large blank areas – the continuity loss pushes other cells to fill those areas appropriately or at least to be aware of them.

We note that continuity loss has a dependence on identifying which cells are adjacent in ground truth. During training, we use the ground truth ordering of cells (via their span indices) to determine adjacency relationships for computing L_{cl}. In inference, the model doesn't explicitly use L_{cl}; rather, it has learned a bias towards predicting adjacent cells with touching boundaries.

Overlap Loss: Despite alignment and continuity constraints, the model could still output overlapping cells (especially if it's unsure of boundaries or in cases of spanning where one cell might overrun into another's space). We introduce an **overlap loss** L_{ol} to penalize any overlapping area between every pair of predicted cells. Instead of computing 2D overlap directly (which could zero out if vertical but not horizontal overlap), we found it effective to treat horizontal and vertical overlaps separately as two terms. Let $\mathbb{I}[i \neq j]$ ensure we only consider distinct cell pairs. The overlap loss is:

$$\mathcal{L}_{ol}^{x} = \sum_{i,j} ||(min(X_r^i, X_r^j) - max(X_l^i, X_l^j))||_1^1 \cdot \mathbb{I}(i! = j)$$

$$\mathcal{L}_{ol}^{y} = \sum_{i,j} ||(min(Y_b^i, Y_b^j) - max(Y_t^i, Y_t^j))||_1^1 \cdot \mathbb{I}(i! = j)$$

$$\mathcal{L}_{ol} = \mathcal{L}_{ol}^{x} + \mathcal{L}_{ol}^{y} \tag{4.4}$$

This effectively adds an L_1 penalty on any positive overlap between boxes in the horizontal and vertical directions. By penalizing even small overlaps, the model learns to output cells that either just touch or have a slight gap (which continuity loss discourages) rather than overlapping. Combined with a Non-Maximum Suppression (NMS) threshold that we apply at inference (we use a relatively high NMS threshold because our model tries not to overlap anyway), this ensures each pixel in the table image is assigned to at most one cell.

The structural losses (L_{al}, L_{cl}, L_{ol}) function as additional regularizers during training. We weight them and add to the overall loss. We found that introducing these losses gradually (e.g., after a certain number of epochs once the detector has reasonable proposals) or using a lower weight initially can stabilize training, but ultimately we treat their weights $\lambda_{al}, \lambda_{cl}, \lambda_{ol}$ as trainable parameters that the min-max optimizer will adjust (discussed next).

4.3.2.1 Trainable Loss Weighting (Min-Max Optimization):

Different regions of a table might require different emphasis on alignment vs. continuity vs. overlap. For example, a very sparse table might benefit less from overlap penalty (since overlaps are unlikely) but more from alignment; a very dense table might need continuity enforcement strongly. Rather than tuning the loss weights globally, we allow the network to learn weights via a min-max formulation. Specifically, we introduce trainable scalars $W_{al}, W_{cl}, W_{ol}^{x}, W_{ol}^{y}$ associated with each structural loss component (with the constraint $W_{al} + W_{cl} + W_{ol}^{x} + W_{ol}^{y} = 1$ to avoid trivial solutions). During training, we alternate between optimizing the model parameters θ_m (minimizing the weighted loss) and optimizing the weights W (maximizing the loss, i.e., finding which constraint is most violated) in a manner inspired by generative adversarial training:

$$\min_{\theta_m} \max_{W}; ; L_{det}(\theta_m) + W_{al} L_{al} + W_{cl} L_{cl} + W_{ol}^{x} L_{ol}^{x} + W_{ol}^{y} L_{ol}^{y}, \tag{4.5}$$

subject to $W_{al} + W_{cl} + W_{ol}^{x} + W_{ol}^{y} = 1$ and $W_{\cdot} \geq 0$. In practice, we implement this by making W logits that are passed through a softmax to ensure non-negativity and sum-to-1, and treating the W parameters as part of the model that get updated with reversed sign on their gradient (since we want to maximize w.r.t. W while minimizing w.r.t. θ_m). The effect is that W will increase for loss terms that are relatively large (indicating the model is struggling with that aspect, so the training focuses more on it), and decrease for terms that are already small (indicating that constraint is largely satisfied). This dynamic weighting scheme improved convergence speed and final accuracy in our experiments.

Concretely, the combined objective for the detection stage becomes:

$$L_{\text{top-down}} = L_{\text{cls}} + L_{\text{bbox}} + L_{\text{GIoU}}; +; W_{al}L_{al} + W_{cl}L_{cl} + W_{ol}^{x}L_{ol}^{x} + W_{ol}^{y}L_{ol}^{y}, , \tag{4.6}$$

with W values being adjusted during training via the min-max schedule. By the end of training, we observed that the loss weights for alignment and continuity tend to be relatively high, indicating the model found those constraints particularly important, especially for datasets like FinTabNet where many cells must align perfectly. The overlap loss weight often remained moderate, as the model naturally avoids overlaps once alignment and continuity are enforced.

In summary, the Top-Down module produces a set of predicted cell boxes that adhere closely to a consistent grid structure. Next, we describe how we determine the adjacency between these cells.

4.3.3 Bottom-Up Structure Recognition (Adjacency Prediction)

After detecting the cells, TabStruct-Net V2 performs structure recognition by determining how these cells are arranged in the table grid. We formulate this as a graph problem: each cell is a node, and we aim to predict edges between cells that are adjacent in the table (above-below or left-right relationships). Instead of directly predicting discrete row/column span indices, we predict adjacency matrices, which is equivalent to reconstructing the table's grid graph. In TabStruct-Net V2, we create four $n \times n$ **rectilinear adjacency matrices** $\mathbf{M}^L, \mathbf{M}^R, \mathbf{M}^T, \mathbf{M}^B \in 0, 1^{n \times n}$, where n is the number of predicted cells. For any two cells i and j: $M_{i,j}^L = 1$ if cell j is immediately to the left of cell i in the same row, $M_{i,j}^R = 1$ if j is immediately to the right of i in the same row, $M_{i,j}^T = 1$ if j is immediately above i in the same column, $M_{i,j}^B = 1$ if j is immediately below i in the same column. These matrices are sparse and satisfy logical relationships (e.g., M^L is essentially the transpose of M^R, and similarly M^T vs M^B; also they are acyclic because they represent grid ordering). The task is to predict these matrices from the image (implicitly from the cell features).

We take a **self-attention approach** for adjacency prediction. For each detected cell, we have a corresponding feature vector from the Top-Down decoder (the output embedding of the DETR query for that cell). Let us denote by $z_i \in \mathbb{R}^d$ the feature for cell i. We form four separate self-attention layers (one for each adjacency direction) that take the set $z_i i = 1^n$ as input queries and keys. Each such layer computes an attention score between every pair of cells:

$$\text{score}_{i,j}^{(\text{dir})} = \frac{1}{\sqrt{d}} \left(W_{\text{dir}}^{Q} z_i \right) \cdot \left(W_{\text{dir}}^{K} z_j \right), \tag{4.7}$$

for $dir \in L, R, T, B$, where $W^Q dir, W_{dir}^K$ are learnable projection matrices for queries and keys in the given direction's attention head. We do *not* apply a softmax to these scores, because we want raw adjacency logits (we found that omitting the softmax yields better results, treating it akin to a linear layer

that computes pairwise affinities, as also noted by other works). We interpret the resulting scores as the weights of adjacency: a higher score indicates a stronger likelihood that cell j is adjacent to cell i in the specified direction.

We then apply a sigmoid to each score to convert it to a probability (since adjacency is binary). The adjacency loss $\mathcal{L}adj$ is computed as a binary focal loss (since the class imbalance between adjacent vs. non-adjacent pairs can be high for large n) between the predicted adjacency probabilities and the ground-truth adjacency matrices. Concretely, for each direction matrix $\mathbf{M}^{dir}$ we have:

$$L_{\text{adj}}^{\text{dir}} = \frac{1}{n^2} \sum_{i,j} \text{FocalLoss}\left(\sigma\left(\text{score}_{i,j}^{(\text{dir})}\right), M_{i,j}^{\text{dir}}\right), \tag{4.8}$$

where $\sigma(\cdot)$ is the sigmoid function. We sum the losses for all four directions:

$$L\text{adj} = L^L adj + L^R adj + L^T adj + L^B adj, . \tag{4.9}$$

Our adjacency prediction module eliminates the need for iterative graph propagation or expensive combinatorial algorithms. It essentially learns a similarity metric in feature space that correlates with actual spatial adjacency in the table. Because the DETR queries attend to relevant regions of the image when detecting cells, their output embeddings z_i carry information about that cell's context (for example, a cell's embedding will encode something about its neighbors because the attention and deformable sampling have seen nearby features). Thus, it is feasible for a simple self-attention to distinguish which cells should connect.

One improvement we apply is to include **positional encoding based on cell coordinates** when computing adjacency. Specifically, we augment the cell feature with a low-dimensional encoding of its normalized bounding box coordinates so that the self-attention can use spatial information to decide adjacency (e.g., cells that are far apart should likely not be adjacent). We found this boosts precision on tricky cases. In practice, we add the learned positional embedding of each cell (projected from its box coordinates) to the queries and keys in the adjacency attention layers. This guides the adjacency head to prefer linking cells that are geometrically aligned and proximate.

Importantly, by predicting separate left/right/top/bottom adjacencies, we inherently allow for multi-row and multi-column spanning cells. For example, if a cell spans multiple rows, it will have multiple other cells with $M^L = 1$ or $M^R = 1$ relations (one for each row it spans) but only one M^T above it (the cell at its top) and one M^B below it (the cell at its bottom). Our adjacency formulation can capture that naturally.

During inference, we take the sigmoid outputs for adjacency and threshold them (we use a threshold of 0.5 by default; since we train with focal loss, outputs are usually confident 0 or 1). We also ensure consistency: for any predicted link $i \xrightarrow{Right} j$, we enforce $j \xrightarrow{Left} i$ (we take the logical AND of the two predictions or simply symmetrize by thresholding both). Similarly for top/bottom. In practice, the model learns to output consistent adjacencies most of the time.

Our Bottom-Up module design is much simpler and more efficient than the graph-based approach in TabStruct-Net V1 [1]. In TabStruct-Net V1, a dynamic GNN was used which involved sampling k nearest neighbor cells for each cell and performing message passing to classify edges; this was slow and did not easily scale to more than a few dozens of cells without subsampling. In contrast, our self-attention adjacency is essentially a single matrix multiplication of size $n \times n$ (with some linear projections), which even for $n = 100$ cells is trivial to compute (and scales to much larger n if needed). We completely remove the need for sampling neighbors, and training this adjacency head is straightforward with our joint loss.

Once we have the adjacency matrices, reconstructing the table structure (assigning row/column indices) is done in the post-processing step.

4.3.4 Post-Processing to Reconstruct Table Structure

After obtaining the set of predicted cell bounding boxes and the four adjacency matrices, the final step is to determine each cell's position in the table grid (i.e., its row and column span indices) and produce the structured output (for example, as HTML or a custom XML). We achieve this by graph traversal and sorting, followed by minor geometric adjustments.

First, we treat the predicted adjacency graph as four directed relations (left/right/top/bottom). We build a grid by finding connected components in this adjacency graph and then topologically sorting them. Concretely, we identify the cell(s) with no cell above them (in-degree $= 0$ in the M^T graph) as top row cells, and similarly leftmost cells (in-degree $= 0$ in M^L) as the first column cells. We then perform a row-by-row, column-by-column traversal:

- We start at the top-left cell (the one with no cell above or to its left).
- We move rightward following the M^R links to enumerate the first row of the table. When we can no longer move right (no outgoing M^R), we move down to the next row.
- For moving to the next row, we identify the leftmost cell of that row (one that has no cell to its left, M^L in-degree 0 within that row's component). This should be linked below the leftmost cell of the previous row (via a M^B link). We then traverse that row via M^R links, and so on.

This procedure yields an ordering of cells by rows and columns. If the adjacency graph is perfect, this gives us directly each cell's (SR, SC, ER, EC) indices by simple counting (e.g., the first cell you encounter is start row 1, start col 1; if it has a cell directly below it, then it spans only one row, etc., which we can infer from missing adjacency in certain directions). In practice, we can determine the span of each cell by seeing how many M^R links emanate from it (that plus one is its column span) and how many M^B links (plus one is its row span). For example, if a cell has outgoing M^R links to two other cells (meaning it touches two distinct cells to its right across different rows), that indicates it spans two rows (because it occupied that many rows such that each had a neighbor to the right).

We also handle any inconsistencies: for example, if the adjacency graph predicted that some cell i is both to the left and to the right of another cell j (which shouldn't happen logically), we resolve by using spatial coordinates as tiebreaker (e.g., compare box centroids). In our implementation, we found few such conflicts; when they occurred, it was usually due to spurious adjacency predictions on complex spanning configurations, which we fixed by trusting geometric sorting (e.g., sorting all cells by their top-left coordinates and using that ordering if adjacency fails).

During post-processing, we also incorporate OCR output *if needed* to refine boundaries. In some cases, a spanning cell might have been predicted slightly mis-sized. We use the text tokens (if available from an OCR engine like Tesseract [45]) inside each predicted cell to adjust its edges slightly: for instance, if two adjacent cells have a small gap or overlap, but token bounding boxes suggest a clear separation, we adjust coordinates accordingly. This is a minor heuristic to improve final output quality and ensure that if the structure is used to extract text, the text assignments are correct. This step is similar to what we did in TabStruct-Net V1's post-processing [1], [2], except simplified.

Finally, we output the table in the desired format. Each cell TC_i is output with attributes: SR_i, SC_i, ER_i, and EC_i along with its text content (if we perform an OCR to extract it; otherwise just structure). The overall table can be represented as an HTML table with appropriate rowspan and colspan attributes, or any equivalent structured representation. In our evaluation, we compare against ground truth structures by basically comparing these (SR, SC, ER, EC) tuples for each cell (this is what metrics like TEDS do under the hood, albeit through an edit-distance on HTML). To summarize, the post-processing takes the high-level predictions (cells + adjacency) and reconstructs the table grid. Because our model is designed to produce outputs that already obey alignment and continuity, the assembly is straightforward and largely deterministic. In contrast, many prior works had to include complex reasoning in post-processing to handle missed or extra cells. Our method is simpler: the learning has handled it.

4.4 Implementation Details

Dataset Pre-processing: For the ground truth data, tight bounding boxes around the cells' content are provided as per previous works [3], [24], [48]. We apply an additional pre-processing step to this ground truth data, building on the approach from [1]. This step ensures that the bounding boxes of cells within the same row are aligned vertically while those in the same column are aligned horizontally.

Data Augmentation: Fig. 4.2 illustrates that FinTabNet [4] and SciTSR [24] differ significantly in visual characteristics due to their origins in business documents and scientific articles, respectively. This disparity leads to uneven predictive performance across domains. To mitigate this, we propose a heuristic-driven data augmentation strategy. Business tables often have the first row and column as headers with numeric values filling dense cells, contrasting with scientific tables. Notably, some scientific tables share similar header structures. We exploit this by vertically appending the same table two or three times (depending on the aspect ratio) while keeping a single header row, ensuring visual consistency with business tables.

Table Splitting at Inference: Before feeding the input image to TabStruct-Net V2, we run OCR to locate token bounding boxes. We identify tokens with overlapping Y regions using an interval tree data structure, thereby identifying several lines in the table. We split the table image into k splits such that each split has no more than ten lines, including two from the previous split and two from the next split. We feed each split to the model and merge the tables using the overlapping regions between each split. It effectively handles recognizing structure for dense tables.

Reproducibility: Our model has been trained and evaluated with table images scaled to a fixed size of 800×800 while maintaining the original aspect ratio as the input. While training, cell-level bounding boxes and row and column adjacency matrices (prepared from start-row, start-column, end-row, and end-column indices) are used as the ground truth. We use M1 ULTRA with 64 GB unified memory for our experiments and a batch size 2. We employ a single attention head for each self-attention module to predict adjacency weights.

4.5 Evaluation

4.5.1 Quantitative Results and Analysis

We utilize the FinTabNet [4] (FTN), SciTSR [24], and our curated datasets for training. TabStruct-Net V2 is then evaluated on the FinTabNet and PubTabNet [50] (PTN) datasets using the Tree Edit Distance Similarity (TEDS) [48] (IC-13), SciTSR, ICDAR-2019 [41] (cTDaR), TUCD, and UNLV benchmarks, we apply the F1 measure based on Cell Adjacency Relations (CAR-F1) as defined in [48]. To address alignment and continuity constraints imposed by the model, we evenly distribute horizontal

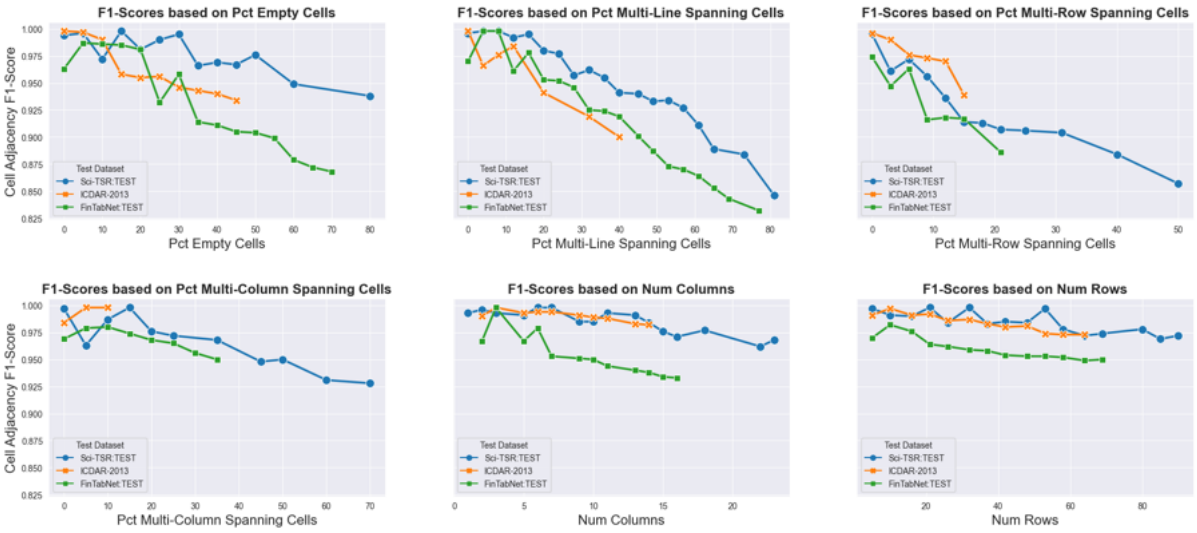


Figure 4.8: Presents F1-Score performance of TabStruct-Net V2 across the ICDAR-2013 (blue), FinTabNet-Test (orange), and SciTSR-Test (green) datasets for each visual characteristic. Each point represents the average F1-Score achieved when trained on the curated dataset. For a clearer view, zoom in.

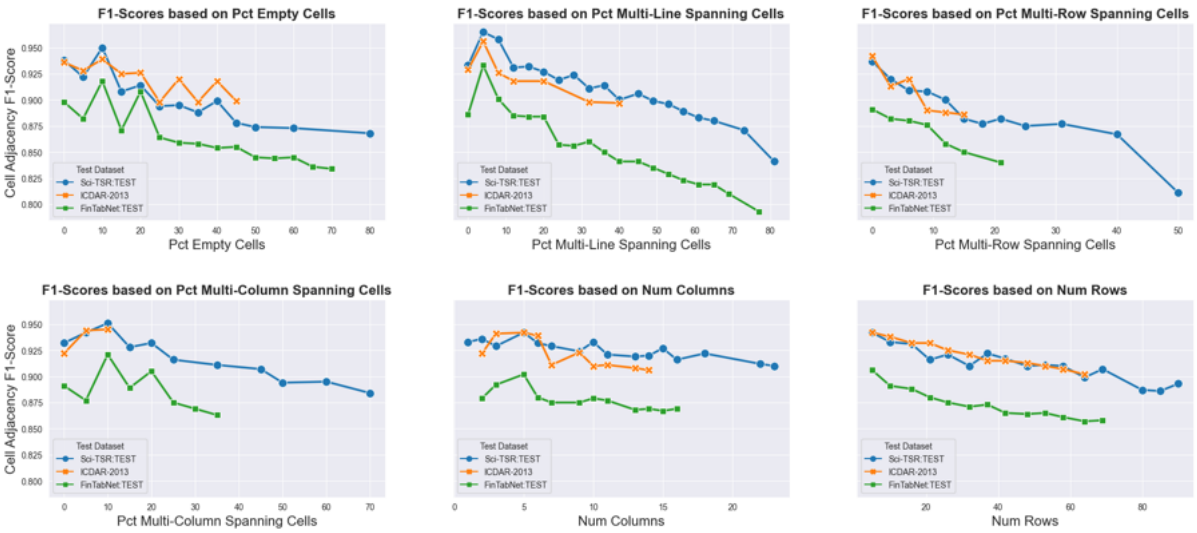


Figure 4.9: Presents F1-Score performance of original TabStruct-Net [1] across ICDAR-2013 (blue), FinTabNet-Test (orange), and SciTSR-Test (green) datasets for each characteristic. Each point represents the average F1-Score achieved when trained on the curated dataset. For a clearer view, zoom in.

	CAR-F1					Struct-TEDS		TEDS		AP_{50}	
Method	*IC-13*	*SciTSR*	*cTDaR*	*TUCD*	*UNLV*	*FTN*	*PTN*	*FTN*	*PTN*	*FTN*	*PTN*
GraphTSR [24]	87.2	95.3	-	-	-	-	-	-	-	-	-
SPLERGE [18]	95.0	92.6	-	-	-	-	-	-	-	-	-
LGPMA [67]	95.3	98.8	-	-	-	-	96.7	-	94.6	-	-
TSRFormer [69]	-	**99.6**	-	-	-	-	97.5	-	-	-	-
GTE [4]	93.5	-	45.9	-	-	91.0	93.0	-	-	-	-
TableFormer [68]	-	-	-	-	-	96.8	96.8	-	93.6	-	82.1
TabStruct-Net [1]	90.6	92.0	58.3	-	83.9	-	-	-	90.1	-	-
TOD-NET + TSR-Net [2]	97.0	93.0	77.0	93.0	-	-	-	-	-	-	-
VAST [149]	96.5	99.5	58.6	-	-	98.6	97.2	**98.2**	96.3	-	94.8
NCGM [70]	98.8	98.8	85.3	-	-	-	95.4	-	-	-	-
LORE [144]	98.9	98.7	88.3	-	-	-	**98.1**	-	-	-	-
GridFormer [150]	-	99.3	-	-	-	98.6	97.0	-	95.8	-	-
TabStruct-Net V2	**99.0**	98.9	**89.1**	-	-	97.2	98.0	96.5	**96.4**	**94.3**	**95.1**

Table 4.1: Presents comparison using CAR-F1 scores at IoU=0.5 on the IC-13, SciTSR, and cTDaR datasets, and S-TEDS and TEDS metrics on the FTN and PTN datasets. To ensure fairness, the training and testing environments for each dataset remain consistent across all methods. As TabStruct-Net V2 generates rectangular bounding boxes, it is not well-suited for handling misaligned or curved tables. Consequently, we do not include a comparison on the WTW dataset.

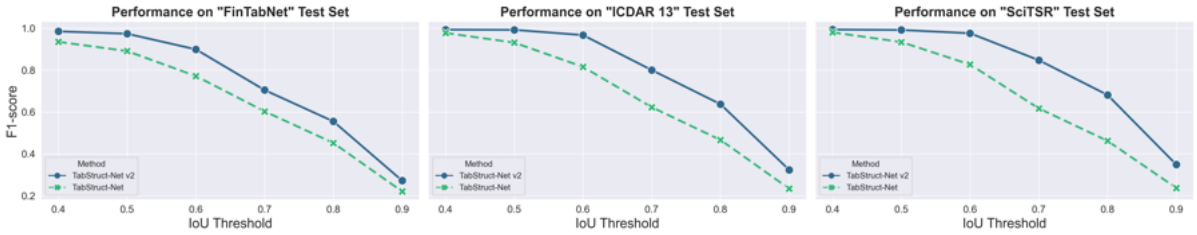


Figure 4.10: Comparison of TabStruct-Net V1 vs TabStruct-Net V2 on varying IoU thresholds on FinTabNet-Test, ICDAR-13 and SciTSR-Test datasets when trained on the same curated dataset.

and vertical gaps between neighboring cells and adjust their boundaries to ensure proper alignment. All evaluations consistently apply an IoU threshold of 0.5. As shown in Table 5.1, our approach outperforms all previous methods on the ICDAR-2013 and cTDaR datasets, surpassing the best-known CAR-F1 scores by 0.1% and 0.8%, respectively, while establishing new benchmarks on the TUCD and UNLV datasets. For the ICDAR-2013 dataset, we follow the evaluation strategy used by [1], [70], [144], utilizing a subset of the dataset for fine-tuning and testing. On the cTDaR dataset, we compute results using an IoU threshold of 0.5, consistent with the strong baseline GTE. Table 5.1 presents these metrics as weighted average F1 scores. Additionally, TabStruct-Net V2 achieves new state-of-the-art benchmarks for S-TEDS and TEDS on PubTabNet. We also report cell detection performance (AP_{50}) as average precision (AP) at an IoU threshold of 0.5 for the PubTabNet and FinTabNet datasets.

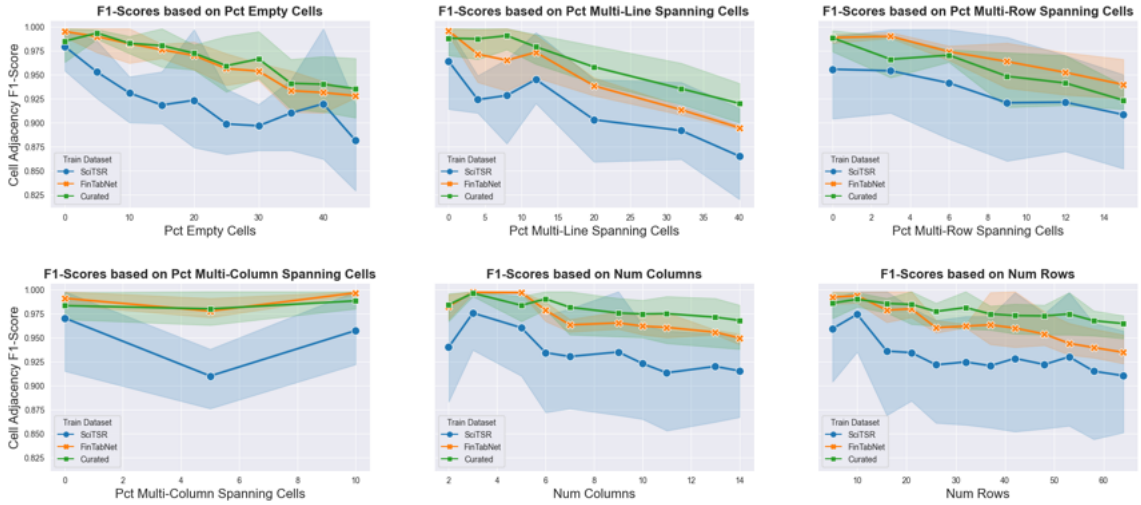


Figure 4.11: Shows the impact of training datasets averaged across the ICDAR-2013, FinTabNet-Test, and SciTSR-Test datasets. Each subgraph corresponds to a specific visual criterion, with lines representing F1-Score distributions for the training datasets: SciTSR (blue), FinTabNet (orange), and Curated (green). The X-axis indicates quantitative complexity, while the Y-axis represents the F1-Score. Shaded regions in each color highlight performance variations across the three test datasets. For consistency, F1-Scores are visualized only for buckets containing a non-zero percentage of table images for each visual criterion. For a clearer view, zoom in.

4.5.2 Analysis based on Visual Characteristics

Fig. 4.3 illustrates the distribution of table images across primary benchmark test datasets for table structure recognition, categorized by visual characteristics. To thoroughly analyze our model's performance, we defined six key visual aspects of a table - percentage of empty cells, percentage of cells spanning multiple lines of text, percentage of cells spanning multiple rows or columns, and number of rows or columns in the table. We then classified each table image in the test datasets into one bucket for each characteristic. All F1-Scores are computed following the standard ICDAR-2019 evaluation criteria using an IoU threshold 0.5.

In Fig. 4.8, each subplot illustrates how the model's F1-Score for predicting correct cell adjacency varies with different visual complexity criteria in tables. The X-axis represents specific table structure characteristics, such as the percentage of empty cells, the percentage of cells spanning multiple lines of text, the percentage of cells spanning multiple rows or columns, and the number of rows or columns in the table. The Y-axis shows the F1-Score, which balances precision and recall, where higher scores indicate better accuracy in reconstructing table structures. The model's performance is compared across three test datasets — SciTSR-Test (blue), ICDAR-2013-Test (orange), and FinTabNet-Test (green). We can observe how increasing complexity affects performance across different domains by analyzing these

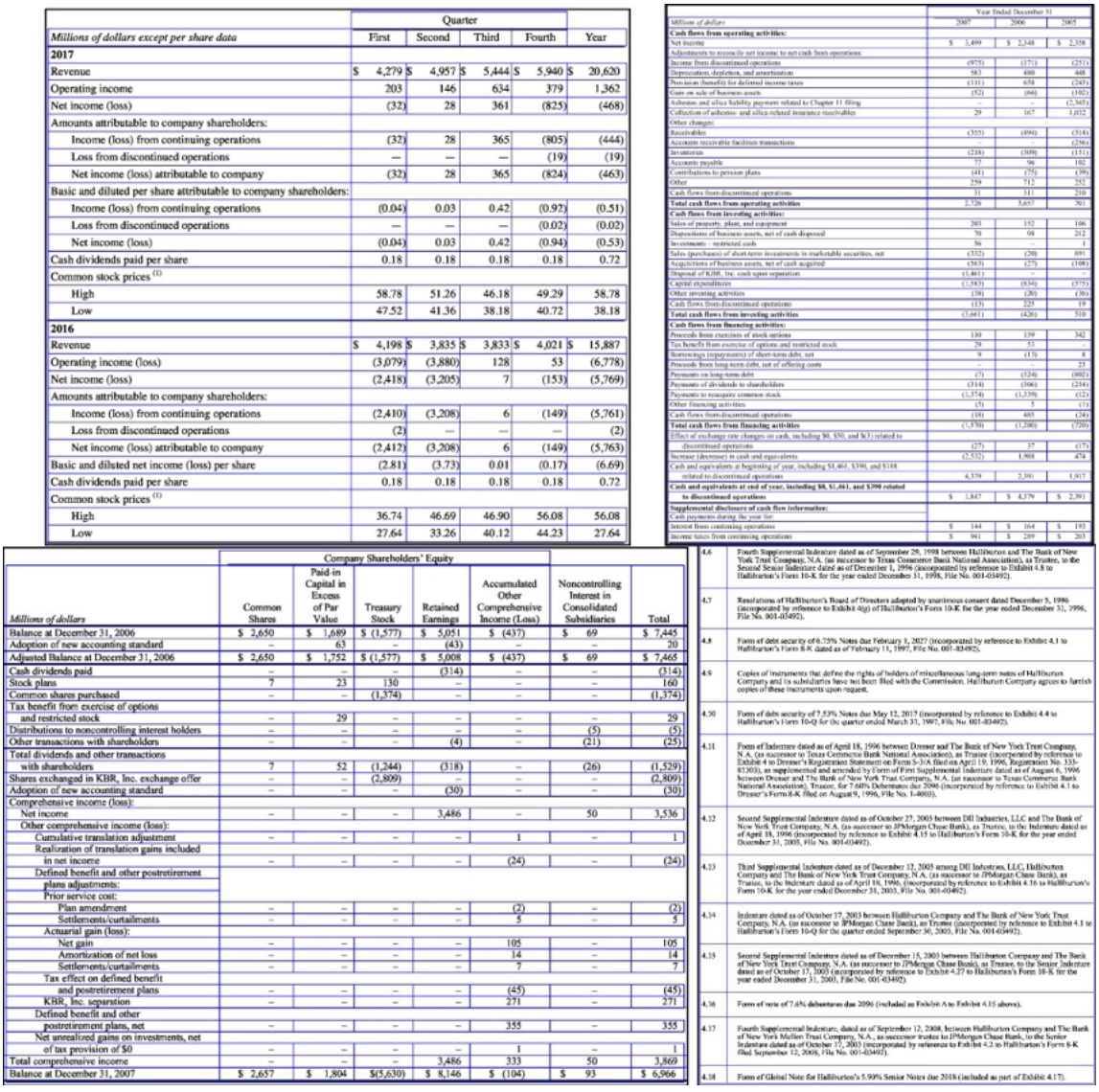
datasets, which vary in table types and distributions.

As complexity increases, the F1-Score generally decreases. When tables contain more empty cells, the model finds it challenging to rely on textual boundaries, reducing its accuracy. Similarly, the structural cues become more ambiguous as cells begin spanning multiple lines, rows, or columns. The model struggles to distinguish where one cell ends and another begins, leading to deteriorating F1-Scores. More extensive tables with more rows and columns also pose more significant challenges — with increasing numbers of cells, there are more opportunities for minor errors that compound into more considerable misunderstandings of the table structure. The differences among the datasets are noteworthy. SciTSR-Test may start from a very high F1-Score but experiences sharper declines as complexity increases. ICDAR-2013-Test may remain more stable, though it also shows a decreasing trend. FinTabNet-Test often presents unique difficulties, possibly because of its tables' more complex or varied nature, resulting in lower F1-Scores when facing increased complexity.

Overall, the collective insight from these graphs is that as tables become visually more complicated—featuring more empty cells, more spanning cells, or larger dimensions — the model's ability to reconstruct their structure accurately diminishes. This understanding points toward the need for more robust training strategies, data augmentation, or specialized modeling techniques to handle real-world tabular data's diversity and complexity. Such an analysis is necessary because the distribution of images belonging to the more complex categories — such as tables with a high percentage of spanning cells or empty cells — is often sparse compared to the more straightforward, more typical cases (see Figs. 4.2 and 4.3). When most training and test data represent straightforward table layouts, the resulting models may appear robust at first glance, even though they have not been adequately tested on more challenging examples. By stratifying and visualizing performance across these increasingly complex buckets, researchers can highlight blind spots and degradation in the model's capabilities. It ensures a more honest assessment of real-world performance, prompts targeted improvements, and guides efforts to balance better or augment training data to represent the full spectrum of potential table structures.

4.5.3 Comparison with TabStruct-Net [1]

A comparative analysis between TabStruct-Net V2 (as shown in 4.8) and its predecessor (shown in 4.9) reveals a significant leap in performance and robustness, particularly as table complexity increases. While both models experience some performance degradation on more challenging tables, the original TabStruct-Net shows a pronounced vulnerability; its F1-score drops sharply when encountering higher percentages of multi-row, multi-column, and multi-line spanning cells. This degradation is also apparent in tables with many empty cells or a high density of rows and columns. In stark contrast, TabStruct-Net V2 demonstrates remarkable resilience across these same criteria. Its performance curves are substantially flatter, indicating the ability to maintain a high F1-score even when faced with complex layouts or



	Quarter				
Millions of dollars except per share data	First	Second	Third	Fourth	Year
2017					
Revenue	$ 4,279	$ 4,957	$ 5,444	$ 5,940	$ 20,620
Operating income	203	146	634	379	1,362
Net income (loss)	(32)	28	361	(825)	(468)
Amounts attributable to company shareholders:					
Income (loss) from continuing operations	(32)	28	365	(805)	(444)
Loss from discontinued operations	—	—	—	(19)	(19)
Net income (loss) attributable to company	(32)	28	365	(824)	(463)
Basic and diluted per share attributable to company shareholders:					
Income (loss) from continuing operations	(0.04)	0.03	0.42	(0.92)	(0.51)
Loss from discontinued operations	—	—	—	(0.02)	(0.02)
Net income (loss)	(0.04)	0.03	0.42	(0.94)	(0.53)
Cash dividends paid per share	0.18	0.18	0.18	0.18	0.72
Common stock prices [1]					
High	58.78	51.26	46.18	49.29	58.78
Low	47.52	41.36	38.18	40.72	38.18
2016					
Revenue	$ 4,198	$ 3,835	$ 3,833	$ 4,021	$ 15,887
Operating income (loss)	(3,079)	(3,880)	128	53	(6,778)
Net income (loss)	(2,418)	(3,205)	7	(153)	(5,769)
Amounts attributable to company shareholders:					
Income (loss) from continuing operations	(2,410)	(3,208)	6	(149)	(5,761)
Loss from discontinued operations	(2)	—	—	—	(2)
Net income (loss) attributable to company	(2,412)	(3,208)	6	(149)	(5,763)
Basic and diluted net income (loss) per share	(2.81)	(3.73)	0.01	(0.17)	(6.69)
Cash dividends paid per share	0.18	0.18	0.18	0.18	0.72
Common stock prices [1]					
High	36.74	46.69	46.90	56.08	56.08
Low	27.64	33.26	40.12	44.23	27.64

	Year Ended December 31		
Millions of dollars	2007	2006	2005
Cash flows from operating activities:			
Net income	$ 3,499	$ 2,348	$ 2,358
Adjustments to reconcile net income to net cash from operations:			
Income from discontinued operations	(975)	(171)	(251)
Depreciation, depletion, and amortization	583	480	448
Provision (benefit) for deferred income taxes	(111)	658	(243)
Gain on sale of business assets	(52)	(66)	(102)
Asbestos and silica liability payment related to Chapter 11 filing	–	–	(2,345)
Collection of asbestos- and silica-related insurance receivables	29	167	1,032
Other changes:			
Receivables	(355)	(494)	(314)
Accounts receivable facilities transactions	–	–	(296)
Inventories	(218)	(309)	(151)
Accounts payable	77	96	102
Contributions to pension plans	(41)	(75)	(39)
Other	259	312	252
Cash flows from discontinued operations	31	311	210
Total cash flows from operating activities	2,726	3,657	701
Cash flows from investing activities:			
Sales of property, plant, and equipment	203	152	106
Dispositions of business assets, net of cash disposed	70	98	212
Investments – restricted cash	56	–	1
Sales (purchases) of short-term investments in marketable securities, net	(332)	(20)	891
Acquisitions of business assets, net of cash acquired	(563)	(27)	(108)
Disposal of KBR, Inc. cash upon separation	(1,461)	–	–
Capital expenditures	(1,583)	(834)	(575)
Other investing activities	(38)	(20)	(36)
Cash flows from discontinued operations	(13)	225	19
Total cash flows from investing activities	(3,661)	(426)	519
Cash flows from financing activities:			
Proceeds from exercises of stock options	189	159	342
Tax benefit from exercise of options and restricted stock	29	53	–
Borrowings (repayments) of short-term debt, net	9	(13)	8
Proceeds from long-term debt, net of offering costs			23
Payments on long-term debt	(7)	(324)	(882)
Payments of dividends to shareholders	(314)	(306)	(284)
Payments to reacquire common stock	(1,374)	(1,339)	(12)
Other financing activities	(5)	5	(1)
Cash flows from discontinued operations	(18)	485	(34)
Total cash flows from financing activities	(1,570)	(1,280)	(720)
Effect of exchange rate changes on cash, including $0, $50, and $(3) related to discontinued operations	(27)	37	(17)
Increase (decrease) in cash and equivalents	(2,532)	1,988	474
Cash and equivalents at beginning of year, including $1,461, $390, and $188 related to discontinued operations	4,379	2,391	1,917
Cash and equivalents at end of year, including $0, $1,461, and $390 related to discontinued operations	$ 1,847	$ 4,379	$ 2,391
Supplemental disclosure of cash flow information:			
Cash payments during the year for:			
Interest from continuing operations	$ 144	$ 164	$ 193
Income taxes from continuing operations	$ 961	$ 289	$ 203

	Company Shareholders' Equity						
Millions of dollars	Common Shares	Paid-in Capital in Excess of Par Value	Treasury Stock	Retained Earnings	Accumulated Other Comprehensive Income (Loss)	Noncontrolling Interest in Consolidated Subsidiaries	Total
Balance at December 31, 2006	$ 2,650	$ 1,689	$ (1,577)	$ 5,051	$ (437)	$ 69	$ 7,445
Adoption of new accounting standard	–	63	–	(43)	–	–	20
Adjusted Balance at December 31, 2006	$ 2,650	$ 1,752	$ (1,577)	$ 5,008	$ (437)	$ 69	$ 7,465
Cash dividends paid	–	–	–	(314)	–	–	(314)
Stock plans	7	23	130	–	–	–	160
Common shares purchased	–	–	(1,374)	–	–	–	(1,374)
Tax benefit from exercise of options and restricted stock	–	29	–	–	–	–	29
Distributions to noncontrolling interest holders	–	–	–	–	–	(5)	(5)
Other transactions with shareholders	–	–	–	(4)	–	(21)	(25)
Total dividends and other transactions with shareholders	7	52	(1,244)	(318)	–	(26)	(1,529)
Shares exchanged in KBR, Inc. exchange offer	–	–	(2,809)	–	–	–	(2,809)
Adoption of new accounting standard	–	–	–	(30)	–	–	(30)
Comprehensive income (loss):							
Net income	–	–	–	3,486	–	50	3,536
Other comprehensive income (loss):							
Cumulative translation adjustment	–	–	–	–	1	–	1
Realization of translation gains included in net income	–	–	–	–	(24)	–	(24)
Defined benefit and other postretirement plans adjustments:							
Prior service cost:							
Plan amendment	–	–	–	–	(2)	–	(2)
Settlements/curtailments	–	–	–	–	5	–	5
Actuarial gain (loss):							
Net gain	–	–	–	–	105	–	105
Amortization of net loss	–	–	–	–	14	–	14
Settlements/curtailments	–	–	–	–	7	–	7
Tax effect on defined benefit and postretirement plans	–	–	–	–	(45)	–	(45)
KBR, Inc. separation	–	–	–	–	271	–	271
Defined benefit and other postretirement plans, net	–	–	–	–	355	–	355
Net unrealized gains on investments, net of tax provision of $0	–	–	–	–	1	–	1
Total comprehensive income	–	–	–	3,486	333	50	3,869
Balance at December 31, 2007	$ 2,657	$ 1,804	$(5,630)	$ 8,146	$ (104)	$ 93	$ 6,966

4.6	Fourth Supplemental Indenture dated as of September 29, 1998 between Halliburton and The Bank of New York Trust Company, N.A. (as successor to Texas Commerce Bank National Association), as Trustee, to the Second Senior Indenture dated as of December 1, 1994 (incorporated by reference to Exhibit 4.8 to Halliburton's Form 10-K for the year ended December 31, 1998, File No. 001-03492).
4.7	Resolutions of Halliburton's Board of Directors adopted by unanimous consent dated December 5, 1996 (incorporated by reference to Exhibit 4(g) of Halliburton's Form 10-K for the year ended December 31, 1996, File No. 001-03492).
4.8	Form of debt security of 6.75% Notes due February 1, 2027 (incorporated by reference to Exhibit 4.1 to Halliburton's Form 8-K dated as of February 11, 1997, File No. 001-03492).
4.9	Copies of instruments that define the rights of holders of miscellaneous long-term notes of Halliburton Company and its subsidiaries have not been filed with the Commission. Halliburton Company agrees to furnish copies of these instruments upon request.
4.10	Form of debt security of 7.53% Notes due May 12, 2017 (incorporated by reference to Exhibit 4.4 to Halliburton's Form 10-Q for the quarter ended March 31, 1997, File No. 001-03492).
4.11	Form of Indenture dated as of April 18, 1996 between Dresser and The Bank of New York Trust Company, N.A. (as successor to Texas Commerce Bank National Association), as Trustee (incorporated by reference to Exhibit 4 to Dresser's Registration Statement on Form S-3/A filed on April 19, 1996, Registration No. 333-81303), as supplemented and amended by Form of First Supplemental Indenture dated as of August 6, 1996 between Dresser and The Bank of New York Trust Company, N.A. (as successor to Texas Commerce Bank National Association), Trustee, for 7.60% Debentures due 2096 (incorporated by reference to Exhibit 4.1 to Dresser's Form 8-K filed on August 9, 1996, File No. 1-4003).
4.12	Second Supplemental Indenture dated as of October 27, 2003 between DII Industries, LLC and The Bank of New York Trust Company, N.A. (as successor to JPMorgan Chase Bank), as Trustee, to the Indenture dated as of April 18, 1996 (incorporated by reference to Exhibit 4.15 to Halliburton's Form 10-K for the year ended December 31, 2003, File No. 001-03492).
4.13	Third Supplemental Indenture dated as of December 12, 2005 among DII Industries, LLC, Halliburton Company and The Bank of New York Trust Company, N.A. (as successor to JPMorgan Chase Bank), as Trustee, to the Indenture dated as of April 18, 1996, (incorporated by reference to Exhibit 4.16 to Halliburton's Form 10-K for the year ended December 31, 2005, File No. 001-03492).
4.14	Indenture dated as of October 17, 2003 between Halliburton Company and The Bank of New York Trust Company, N.A. (as successor to JPMorgan Chase Bank), as Trustee (superseded by reference to Exhibit 4.1 to Halliburton's Form 10-Q for the quarter ended September 30, 2005, File No. 001-03492).
4.15	Second Supplemental Indenture dated as of December 15, 2003 between Halliburton Company and The Bank of New York Trust Company, N.A. (as successor to JPMorgan Chase Bank), as Trustee, to the Senior Indenture dated as of October 17, 2003 (incorporated by reference to Exhibit 4.27 to Halliburton's Form 10-K for the year ended December 31, 2003, File No. 001-03492).
4.16	Form of note of 7.6% debentures due 2096 (included as Exhibit A to Exhibit 4.15 above).
4.17	Fourth Supplemental Indenture, dated as of September 12, 2008, between Halliburton Company and The Bank of New York Mellon Trust Company, N.A., as successor trustee to JPMorgan Chase Bank, to the Senior Indenture dated as of October 17, 2003 (incorporated by reference to Exhibit 4.2 to Halliburton's Form 8-K filed September 12, 2008, File No. 001-03492).
4.18	Form of Global Note for Halliburton's 5.90% Senior Notes due 2018 (included as part of Exhibit 4.17).

Figure 4.12: Sample outputs of TabStruct-Net V2 from Sci-TSR-Test and FinTabNet-Test datasets in varying layouts and visual characteristics.

dense structures, showcasing its superior generalization capabilities.

These nuanced improvements are not arbitrary but are the direct outcome of targeted architectural and strategic enhancements in TabStruct-Net V2. The model's enhanced ability to correctly interpret tables with various spanning and empty cells stems from the our novel The Hierarchical Local-Attention Vision Transformer (HLViT) backbone for learning latent representations, replacement of the inefficient, sampling-based DGCNN with a more scalable self-attention mechanism for predicting cell adjacencies, and the implementation of a split-and-merge strategy for inference. This allows the model to effectively process extremely large tables that would overwhelm its predecessor, thus addressing a key failure point and resulting in a fundamentally more robust and accurate solution for table structure recognition.

Further, as illustrated in the figure 4.10, we perform granular analysis of the models' performance across a spectrum of IoU thresholds, from a lenient 0.4 to a highly stringent 0.9, to get profound insights into their localization precision. The primary observation is the unequivocal and consistent outperformance of TabStruct-Net V2 over its predecessor across all three benchmark datasets: FinTabNet, ICDAR-2013, and SciTSR. This is not a trivial improvement; it points to a fundamental enhancement in the model's ability to accurately delineate cell boundaries. The most telling nuance lies in the widening performance gap as the IoU threshold increases. For example, on the SciTSR dataset, while both models perform well at an IoU of 0.5, the original TabStruct-Net's F1-score plummets to below 0.7 at an IoU of 0.8. In contrast, TabStruct-Net V2 maintains an F1-score well above 0.8 at the same stringent level.

This trend signifies that the original model, while capable of approximate localization, lacks the precision required for high-fidelity reconstruction, a weakness that becomes starkly evident under strict evaluation. TabStruct-Net V2's more graceful degradation curve is a direct testament to the efficacy of its architectural improvements. The HLViT backbone excels at extracting fine-grained features from high-resolution images, while the refined alignment and continuity loss functions explicitly guide the model to predict well-aligned cell coordinates. Consequently, TabStruct-Net V2 achieves a fundamentally higher level of precision, confirming that its advancements translate into tangible, robust, and highly accurate cell detection that holds up even under the most demanding evaluation criteria.

4.5.4 Comparison with Commercial and Open-Source VLMs

Table 4.2 summarises representative TEDS-Struct scores on FinTabNet and SciTSR for the TabStructNet family alongside leading commercial and open-source VLMs. Scores for VLMs are reported under zero-shot (‡) conditions where available; scores for specialist models use task-specific training throughout.

The data in Table 4.2 yield three clear findings. First, on **FinTabNet**, TabStructNet-lineage models outperform every zero-shot VLM by a margin of 13 TEDS-Struct points, with even the privacy-constrained TabGuard (operating on content-masked images) achieving state-of-the-art performance.

Table 4.2: TEDS-Struct (%) on FinTabNet and SciTSR. **Bold**: best per group. ‡ = zero-shot; ⋆ = optimal train dataset; † = content-masked (privacy) setting; – = not reported. Reported results are obtained for VLMs in optimal prompt configurations reported experimental setups [151]–[153].

Method	**FinTabNet**		**SciTSR**		**Venue / Year**
	TEDS-S (⋆)	TEDS-S (‡)	TEDS-S (⋆)	TEDS-S (‡)	
Our Specialist TSR Models					
TabStructNet V2 / TOD-Net [53]	>98.34	–	$\approx$99.0††	–	IJDAR 2025
TabGuard† [47]	**SoTA**†	–	$\approx$99.0††	–	WACV 2025
Commercial VLMs (closed API)					
GPT-4o (ChatGPT)	–	88	–	95	OpenAI 2024
Gemini-2.5-Flash (Google)	–	92	–	**96**	Google 2025
Open-Source VLMs					
Qwen2.5-VL-7B [154]	97	90	97	93.12	Alibaba 2024
Qwen2.5-VL-72B [154]	–	93	–	96	Alibaba 2025
InternVL2-76B [155]	–	91	–	96	Shanghai AI Lab 2024
Phi-3-Vision [101]	–	75	–	83	Microsoft 2024
Llama-3.2-Vision-11B [60]	–	80	–	87	Meta 2024

Second, fine-tuned Qwen2.5-VL-7B narrows this gap to $\sim$2 points, but at the cost of $>100\times$ higher inference latency [151], making it impractical for large-scale document processing. Third, on **SciTSR**, a higher-quality image corpus, commercial VLMs (Gemini-2.5-Flash reaching $\sim$96%) approach but do not surpass specialist models, which operate in the 99.0–99.1% range.

Situations Where the Proposed Methods Perform Better The proposed approaches maintain clear advantages in the following scenarios:

1. **Dense financial tables with complex spanning structures.** FinTabNet tables from S&P 500 annual reports contain multi-level column headers, nested row groups, and many empty cells. The graph-based adjacency formulation in TabStructNet encodes structural constraints (alignment, continuity, and overlap losses) that directly penalise ill-formed cell arrangements. VLMs, which generate HTML autoregressively, lack this constraint and accumulate structural errors across long output sequences [152].

2. **Low-quality or variable-resolution inputs.** The NGTR benchmark [152] demonstrates that the performance gap between VLMs and specialist models widens substantially as image quality degrades: the gap reaches 13.6 TEDS points on low-resolution inputs (vs. 4.4 on SciTSR-quality images). The cell-detection backbone of TabStructNet is explicitly trained on the target distribution and is more robust to such degradation.

3. **High-throughput, production-scale processing.** Specialist TSR models run at $>100\times$ the throughput of Qwen2.5-VL-7B for equivalent or superior accuracy on FinTabNet [151]. For large document repositories (e.g., regulatory filing archives, insurance claims), this throughput advantage is decisive.

4. **Privacy-sensitive deployments.** TabGuard [47] is the only high-accuracy TSR system that provides a formal *content-privacy* guarantee: all cell text is masked locally before the table image is transmitted to the recognition server. Every commercial VLM requires sending unmasked table images to a third-party API, making them unsuitable for confidential financial, legal, or medical document processing under strict data-governance frameworks (e.g., GDPR, HIPAA).

5. **Training data efficiency and domain adaptation.** TabStructNet-lineage models achieve their performance by training solely on a few tens of thousands of domain-specific examples, whereas VLMs require enormous general pre-training plus domain fine-tuning. In practice, organisations can fine-tune specialist models on proprietary in-house table corpora without exposing sensitive data to external cloud services.

Cases Where LLM-based Approaches Complement the Proposed Pipeline Despite their limitations on structural accuracy, VLMs offer capabilities that are orthogonal to—and in several ways complementary to—the TabStructNet family:

1. **Semantic cell content understanding.** The TabStructNet family identifies the *structural layout* of cells (which cells are in the same row or column, where spanning occurs), but does not interpret the *meaning* of cell contents. A downstream VLM or LLM can consume the structured output of TabStructNet and perform semantic tasks such as table question answering, numerical reasoning, or natural-language summarisation—tasks for which VLMs are well-suited.

2. **Zero-shot generalisation to novel table domains.** Specialist TSR models generalise best within the distribution they were trained on. For tables from entirely new document types (e.g., legal contracts, handwritten ledgers, or low-resource-language scientific papers), a VLM can provide a usable structural prediction without any retraining. This zero-shot capability can bootstrap annotation for subsequent fine-tuning of a specialist model.

3. **Handling borderless and ill-structured tables.** Tables without explicit ruling lines, or with strongly rotated or distorted geometries (as in the WTW wild-table dataset), pose difficulties for cell-detection methods that rely on visual boundaries. VLMs trained on diverse document images may perform more robustly in these edge cases, serving as a fallback when specialist model confidence is low.

4. **Post-processing and error correction.** A VLM can act as a *verifier*: given the HTML output of TabStructNet, it can identify structural inconsistencies (e.g., a row that contains fewer cells than

expected) and suggest corrections. This human-in-the-loop or automatic verification step could improve end-to-end accuracy beyond what either system achieves alone.

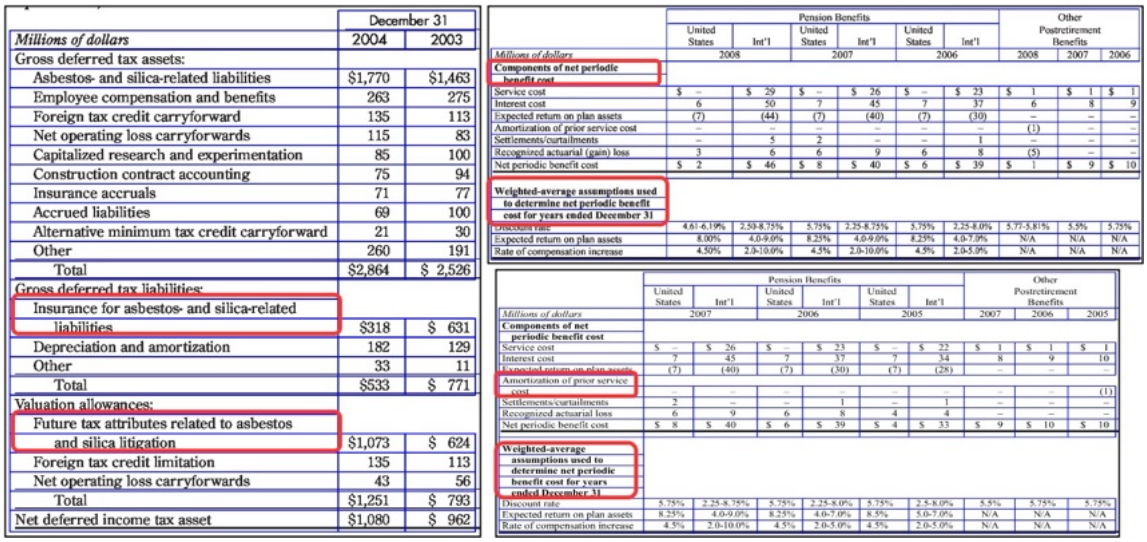
5. **Multimodal document understanding pipelines.** Modern document intelligence systems require understanding of text paragraphs, figures, equations, and tables holistically. A hybrid pipeline—in which TabStructNet handles table cell localisation and adjacency while a VLM handles the broader document context (section headings, cross-references, figure captions)—is a natural architectural decomposition that plays to each component's strengths. This aligns with the design of systems such as MinerU and OmniDocBench, which combine specialist parsers with VLM-based understanding.

In summary, the TabStructNet family and recent LLM-based systems occupy complementary positions in the TSR landscape. Specialist models excel on structural accuracy, throughput, and privacy; VLMs excel on zero-shot versatility, semantic interpretation, and multi-task flexibility. The most promising direction for future work is a *hybrid architecture* that routes structural prediction through a specialist model while delegating semantic enrichment and cross-document reasoning to a VLM, yielding a system that is both structurally precise and semantically rich.

4.5.5 Impact of Training Datasets

Fig. 4.11 demonstrates the curated dataset's effectiveness in evaluating robustness. The presented graphs illustrate how the choice of training datasets influences the robustness and overall performance of a table structure recognition model when confronted with varying levels of visual complexity. Each subplot shows how the F1-Score changes as specific visual criteria become more challenging. For example, the requirements might be the increasing number of empty cells, multi-line cells, multi-row cells, or simply the growing number of table rows and columns. The visual complexity rises as one moves along the horizontal axis in each graph, resulting in noticeable performance degradation. The vertical axis reflects the resulting F1-Scores for three training dataset scenarios — SciTSR, FinTabNet, and Curated, each depicted by a differently colored line and shaded region.

The model trained on the SciTSR dataset (blue line) demonstrates strong performance on more straightforward tables but experiences a sharp decline in F1-Score as table complexity increases. The FinTabNet-trained model (orange line) sustains relatively higher scores than SciTSR as complexity grows but also encounters significant drops under more challenging conditions. In contrast, the model trained on the curated dataset (green line) consistently achieves superior F1-Scores and demonstrates remarkable resilience to the effects of increasing complexity. The curated model maintains robust and stable performance even with challenges such as a higher percentage of empty cells or cells spanning multiple rows and columns. A key finding is the model's strong and stable performance on multi-column cells, even without the highest training percentage for this specific feature. This proves that the strategic selection of holistically complex training samples was more crucial for learning underlying patterns



Millions of dollars	December 31 2004	2003
Gross deferred tax assets:		
Asbestos- and silica-related liabilities	$1,770	$1,463
Employee compensation and benefits	263	275
Foreign tax credit carryforward	135	113
Net operating loss carryforwards	115	83
Capitalized research and experimentation	85	100
Construction contract accounting	75	94
Insurance accruals	71	77
Accrued liabilities	69	100
Alternative minimum tax credit carryforward	21	30
Other	260	191
Total	$2,864	$ 2,526
Gross deferred tax liabilities:		
Insurance for asbestos- and silica-related liabilities	$318	$ 631
Depreciation and amortization	182	129
Other	33	11
Total	$533	$ 771
Valuation allowances:		
Future tax attributes related to asbestos and silica litigation	$1,073	$ 624
Foreign tax credit limitation	135	113
Net operating loss carryforwards	43	56
Total	$1,251	$ 793
Net deferred income tax asset	$1,080	$ 962

	Pension Benefits						Other Postretirement Benefits		
	United States	Int'l	United States	Int'l	United States	Int'l			
Millions of dollars	2008		2007		2006		2008	2007	2006
Components of net periodic benefit cost									
Service cost	$ –	$ 29	$ –	$ 26	$ –	$ 23	$ 1	$ 1	$ 1
Interest cost	6	50	7	45	7	37	6	8	9
Expected return on plan assets	(7)	(44)	(7)	(40)	(7)	(30)	–	–	–
Amortization of prior service cost	–	–	–	–	–	–	(1)	–	–
Settlements/curtailments	–	5	2	–	–	1	–	–	–
Recognized actuarial (gain) loss	3	6	6	9	6	8	(5)	–	–
Net periodic benefit cost	$ 2	$ 46	$ 8	$ 40	$ 6	$ 39	$ 1	$ 9	$ 10
Weighted-average assumptions used to determine net periodic benefit cost for years ended December 31									
Discount rate	4.61-6.19%	2.50-8.75%	5.75%	2.25-8.75%	5.75%	2.25-8.0%	5.77-5.81%	5.5%	5.75%
Expected return on plan assets	8.00%	4.0-9.0%	8.25%	4.0-9.0%	8.25%	4.0-7.9%	N/A	N/A	N/A
Rate of compensation increase	4.50%	2.0-10.0%	4.5%	2.0-10.0%	4.5%	2.0-5.0%	N/A	N/A	N/A

	Pension Benefits						Other Postretirement Benefits		
	United States	Int'l	United States	Int'l	United States	Int'l			
Millions of dollars	2007		2006		2005		2007	2006	2005
Components of net periodic benefit cost									
Service cost	$ –	$ 26	$ –	$ 23	$ –	$ 22	$ 1	$ 1	$ 1
Interest cost	7	45	7	37	7	34	8	9	10
Expected return on plan assets	(7)	(40)	(7)	(30)	(7)	(28)	–	–	–
Amortization of prior service cost	–	–	–	–	–	–	–	–	(1)
Settlements/curtailments	2	–	–	1	–	1	–	–	–
Recognized actuarial loss	6	9	6	8	4	4	–	–	–
Net periodic benefit cost	$ 8	$ 40	$ 6	$ 39	$ 4	$ 33	$ 9	$ 10	$ 10
Weighted-average assumptions used to determine net periodic benefit cost for years ended December 31									
Discount rate	5.75%	2.25-8.75%	5.75%	2.25-8.0%	5.75%	2.5-8.0%	5.5%	5.75%	5.75%
Expected return on plan assets	8.25%	4.0-9.0%	8.25%	4.0-7.0%	8.5%	5.0-7.0%	N/A	N/A	N/A
Rate of compensation increase	4.5%	2.0-10.0%	4.5%	2.0-5.0%	4.5%	2.0-5.0%	N/A	N/A	N/A

Figure 4.13: Sample outputs from TabStruct-Net V2 with errors highlighted in red.

than the sheer volume of specific examples. The shaded regions around each line indicate variability across the three test datasets (ICDAR-2013, FinTabNet-Test, and SciTSR-Test). The curated dataset's narrower and higher-shaded region reflects better and more consistent generalization across these diverse evaluation scenarios.

The results highlight the critical importance of the composition and diversity of training data in developing a generalizable model. While the SciTSR and FinTabNet datasets are valuable, they may not fully capture the wide range of real-world complexities a table recognition system might encounter. Consequently, models trained on these datasets struggle more when confronted with unusual or complex table structures. In contrast, the curated dataset likely includes various table types, layouts, and complexities, preventing the model from becoming overly specialized to a specific table style. This diverse training exposure enables the model to learn more adaptable patterns and handle unseen or challenging scenarios more effectively. Ultimately, the curated dataset fosters resilience, equipping the model to produce consistent and high-quality results across diverse test datasets despite significant table structure and layout variability.

4.6 Ablation Study

Table 4.3 presents an ablation study showcasing incremental improvements in the F1-Score for cell adjacency relations as the baseline Deformable-DETR model is progressively enhanced. The experiments use the curated dataset for training and the ICDAR-2013 test dataset for evaluation. Starting with a baseline F1-Score of 94.1% using a standard ResNet-50 backbone, replacing it with local attention (LA) -aware backbone increases the F1-Score to 96.7%, achieving a 2.6% absolute improvement.

Method	Cell Detection		
	P↑	R↑	F1↑
De-DETR(RESNET-50)	93.5	94.7	94.1
De-DETR(LA)	96.4	97.2	96.7
De-DETR(LA)+AL	97.1	97.6	97.3
De-DETR(LA)+AL+CL	97.8	98.1	97.9
De-DETR(LA)+AL + CL+OL	98.1	98.3	98.2
De-DETR(LA)+AL+CL+OL+LOSSWT	98.2	98.4	98.3
De-DETR(LA)+AL+Data Augmentation+CL+OL+LOSSWT	98.4	98.7	98.5
De-DETR(LA)+AL+Data Augmentation+Splitting Strategy+CL+OL+LOSSWT	**99.0**	**99.1**	**99.0**

Table 4.3: Presents the ablation study for cell detection under various structural constraints applied to the baseline Deformable-DETR. The evaluation follows standard criteria with an IoU threshold of 0.5. LA: denotes the local attention-aware backbone, AL: represents alignment loss, CL: continuity loss, OL: overlapping loss, and LOSSWT refers to loss weights. The model is trained on the SciTSR [24] dataset and evaluated on the FinTabNet-Test dataset [4].

Adding alignment loss (AL) further improves the F1-Score to 97.3% (a 0.6% gain), while incorporating continuity loss (CL) raises it to 97.9% (another 0.6%). Including overlapping loss (OL) marginally boosts the score to 98.2% (a 0.3% increase), and fine-tuning loss weights (Loss WT) results in a slight improvement to 98.3% (0.1%). Beyond loss refinements, data augmentation increases the F1-Score to 98.5% (a 0.2% gain), and introducing a splitting strategy brings the F1-Score to 99.0%, marking a substantial 4.9% improvement over the initial baseline. This progression underscores the cumulative impact of architectural and training-based modifications on enhancing the model's cell detection performance.

4.7 Conclusion

We presented TabStruct-Net V2, a robust anchor-free approach for end-to-end table structure recognition from images. Building on our earlier work, we introduced several key innovations: a Hierarchical Local-Attention Vision Transformer backbone for precise feature extraction on high-resolution tables, a refined top-down cell detection with human-inspired structural losses (alignment, continuity, overlap) that enforce a well-formed table layout, a simplified yet more powerful bottom-up adjacency prediction using self-attention in place of a graph neural network, and a split-and-merge strategy to handle exceptionally large or dense tables.

These advancements yielded a model that significantly outperforms the previous TabStruct-Net V1 across all evaluations. TabStruct-Net V2 achieves higher accuracy in detecting and delineating table cells (with far greater precision at strict overlap criteria) and excels in correctly interpreting complex table structures (setting new state-of-the-art results on benchmarks like SciTSR, PubTabNet, ICDAR-2013, and ICDAR-2019 cTDaR). Particularly, our model maintains performance on challenging cases

such as tables with many empty or spanning cells, thanks to the curated training strategy and the structural losses which guide the model's predictions to be consistent with human expectations of a table. The HLViT backbone proved ideal for table images, capturing both fine details (like small alignment offsets) and broader context in a compute-efficient manner.

Beyond raw numbers, TabStruct-Net V2 produces outputs that are qualitatively superior – reconstructed tables are clean, well-aligned, and usable in downstream applications without further post-correction. Error analysis indicates most remaining mistakes are in extremely tricky scenarios (for instance, distinguishing between two adjacent empty cells vs. one spanning empty cell, in absence of lines or text). These are edge cases that even humans might find ambiguous without additional context.

In terms of efficiency, replacing the prior graph-based structure module with our adjacency self-attention greatly improved inference speed, making TabStruct-Net V2 viable for real-time document processing systems. The elimination of anchor heuristics and the use of an anchor-free DETR framework means our model readily adapts to different image scales and aspect ratios, as well as rotated tables (to a degree – we assume mostly upright tables, but slight rotations are handled).

In conclusion, TabStruct-Net V2 represents a substantial step forward in table structure recognition. It combines the strengths of top-down object detection and bottom-up relational reasoning in a single trainable architecture, guided by principles of how tables are logically organized. The result is a robust, accurate, and scalable solution that sets a new bar on multiple benchmarks. This work not only contributes a high-performing model but also provides a comprehensive analysis framework (with dataset categorization and in-depth evaluation) that can serve as a foundation for future research in structured document analysis.

Moving forward, there are a few avenues to explore. One is incorporating semantics or OCR content more directly into the structure recognition – our current model treats text purely as another visual cue, but integrating language models might help disambiguate cell grouping in very tough cases. Another direction is extending to truly arbitrary table orientations (curved or rotated tables as in the ICDAR-2013 dataset's harder cases or WTW) – while our current method focuses on rectilinear structures, the general approach of combining detection with structural losses could potentially be extended to non-rectangular table detection (e.g., by learning to predict connectivity in a graph of tokens). Finally, applying TabStruct-Net V2's ideas to other structured entities (forms, math formulas, etc.) could be fruitful, given the parallels in needing to detect elements and infer their structural relations.

In summary, TabStruct-Net V2 significantly advances automated table parsing. It achieves high precision and robustness by respecting the inherent geometry and logic of tables, demonstrating how injecting structural knowledge into deep models can pay dividends in performance. We hope this work

will not only be useful for practical document analysis applications (from financial report extraction to archival data digitization) but also inspire further research on structured data extraction using hybrid top-down and bottom-up neural architectures.

Chapter 5

TabGuard: Privacy Preserving and Spatially Aware Table Structure Recognition

5.1 Introduction

Table Structure Recognition (TSR) is a pivotal task in document image analysis and data extraction. It involves converting a table image into a machine-readable format that encodes the table's layout and cell locations in a predefined structure. Formally, the goal is to infer the rows, columns, and spanning relationships of all cells from the visual content of the table. Solutions to TSR have broad applications in digitizing documents from domains like finance, healthcare, law, and scientific publishing. However, tables from these data-sensitive sectors often contain confidential information, raising critical *privacy concerns*. In practical deployments of TSR, deep learning models typically run on powerful GPU servers, requiring clients to send document images to a remote service. Even within an organization's on-premise infrastructure, this client-server paradigm poses risks of unauthorized data access. Sensitive content such as financial figures, personal health records, or legal case data could be exposed to internal threats or logging if transmitted in raw form. Ensuring that table data remains private throughout the recognition process is therefore of paramount importance. This chapter addresses the technical challenge: **Can we perform high-accuracy table structure recognition without revealing any table content to the server?**

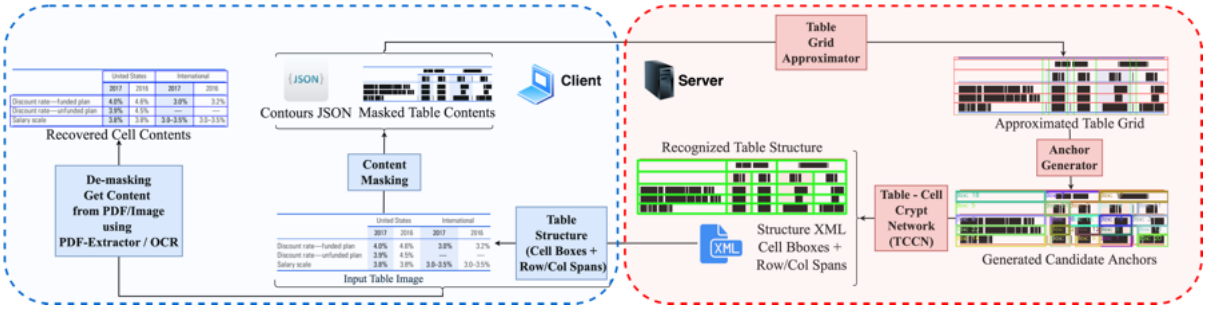


Figure 5.1: Overview of *TabGuard*. Content of the table is only seen by the client. TSR API server sees images with content masked.

TabGuard is our answer to this challenge – a privacy-preserving framework for end-to-end secure TSR. The key idea of TabGuard is to mask all textual content in the table *before* it leaves the client device, so that the server only processes an image with obfuscated cell contents. In our design (illustrated in Figure 5.1), the client first locally applies a content masking algorithm to the table image, rendering textual regions as filled boxes while preserving the graphical structure cues (such as cell boundaries or the spatial gaps corresponding to cell separations). The client also extracts basic layout metadata (e.g. the positions of these text boxes, referred to as "text contours") and sends this along with the masked image to the server. The server, hosting the heavy TSR model, predicts the table's structure (cells and their coordinate indices) from the masked image alone. The predicted structure is then returned to the client, which maps the original content into the detected cells. This architecture ensures that *sensitive content never leaves the client side*: the server operates solely on sanitized images and cannot infer the actual cell contents. The communication overhead is minimal – only a fixed-size image (with content removed) and some lightweight JSON data with contour coordinates are transferred each way. Our implementation supports asynchronous batched requests and easy horizontal scaling on the server, yielding low latency even for high-throughput applications. By enabling privacy-preserving inference in this manner, TabGuard opens the possibility of leveraging powerful cloud-based TSR models for domains like medical records, financial statements, or legal documents, where data privacy is non-negotiable.

Developing an effective TSR method that uses only masked images (with all text removed) is a non-trivial task. Traditional top-down/bottom-up TSR pipelines rely heavily on visual text cues to identify cell boundaries and reading order. In our scenario, the model must decipher the table's structure using primarily the spatial arrangement of blank regions (the masked text blocks) and any ruling lines if present. We adopt a combined *top-down and bottom-up strategy* for this: first infer a rough grid layout of the table (top-down segmentation into rows and columns), then refine and merge these regions to detect actual table cells and their relationships (bottom-up grouping). A core requirement here is accurate detection of each table cell's location, even when cells span multiple rows or columns. However, standard object detection frameworks face significant challenges in this setting. For instance, convolutional detectors like Faster R-CNN [29], [30] use a fixed set of anchor boxes (defined by predetermined sizes, aspect ratios, and strides) as initial cell proposals. These anchors are pruned by a Region Proposal Network (RPN) and Non-Maximal Suppression (NMS), and the final performance heavily depends on having an appropriate anchor that overlaps each true cell. In tables, cells vary drastically in size (from a tiny symbol in a corner to a multi-row merged cell spanning half the table) and aspect ratio (very wide or very tall cells are common). It is practically impossible to cover all such extremes with a small set of predefined anchor shapes. If a true cell does not overlap sufficiently with any anchor, it will be missed (false negative); on the other hand, using too many or too large anchors leads to excessive false positives and NMS conflicts. Indeed, prior studies have noted that objects with extreme scales or aspect ratios tend to be problematic for anchor-based detectors. Table cells in dense tables often trigger exactly these issues: numerous cells packed closely together cause NMS to mistakenly discard legitimate detections,

and cells with extreme dimensions fall outside the anchor grid coverage. Recent one-stage detection approaches based on Transformers, such as DETR [156] and its variants [147], [157], eliminate explicit anchors but have shown poor performance on small or densely packed objects. Their set-based predictions can struggle to attend to every tiny cell when the number of cells varies widely across images. These factors make naïvely applying existing object detectors to masked table images ineffective for high-density tables.

5.1.1 Contributions

TabGuard overcomes these obstacles with a novel two-stage approach that injects strong inductive bias tailored to table structures. First, instead of relying on arbitrary anchor initialization, we leverage the intrinsic **grid alignment** of table data. We approximate the table's underlying grid structure by analyzing the distribution of masked text contours (the blanked-out text regions) in the image. This yields a coarse but comprehensive set of potential row and column dividers, essentially hypothesizing the table's grid lines without seeing actual content. From this approximated grid, we then generate **dynamic anchors** that explicitly cover every possible table cell location. In effect, for each grid cell (the smallest subdivision from the approximated grid), TabGuard considers not just that cell but also every contiguous rectangular merge of up to a fixed number of rows and columns as a candidate anchor. This process produces anchor boxes that tightly span all plausible table cells, including those that extend across multiple rows/columns. Crucially, it guarantees that for each true table cell, *at least one* anchor box will have a high overlap with it, even for cells of extreme shape. By design, this removes the reliance on guesswork for anchor sizes and largely sidesteps the false negative issue caused by anchor misalignment. We call our grid inference module the *Table Grid Approximator (TGA)*, and the resulting set of anchors *grid-cell anchors*. The use of TGA provides an inductive bias that exploits the regular structure of tables, guiding the detector towards aligned row/column boundaries from the start.

With this comprehensive anchor set, the detection of table cells is handed over to our **Table Cell Crypt Network (TCCN)**. TCCN is a specialized object detection network based on the Fast R-CNN paradigm [138], streamlined for table cells on masked images. By using the pre-generated grid-cell anchors, we entirely remove the need for an RPN stage and its associated hyperparameters. Instead, the network directly evaluates the candidate anchors from TGA, significantly simplifying training and improving efficiency. We employ a ResNeXt-101 [158] backbone with a Feature Pyramid Network (FPN) to extract multi-scale features, and apply RoI Align on each anchor box to pool features for that region. The network then classifies each candidate anchor as a table cell or not, and regresses precise bounding-box coordinates for the cell. To refine the quality of predictions and enforce the structural consistency of the table, we augment the standard detection loss with **alignment and continuity losses** as introduced in prior works. The alignment loss encourages all cells in the same row to share the same top and bottom boundaries, and all cells in the same column to share the same left and right boundaries. The continuity loss ensures that adjacent rows and columns line up without gaps – for any two consecutive cells ver-

tically, the bottom of the upper cell should match the top of the lower cell, and likewise horizontally for side-by-side cells. By formulating these as smooth L_1 penalties on the coordinates, TCCN learns to output cells that form a tightly aligned grid, mirroring the true table structure. Once TCCN detects all the table cells (represented as bounding boxes), determining the table's structural hierarchy (which cell is in which row/column and which cells span multiple rows/columns) becomes straightforward. We perform a lightweight **post-processing** step to assign row and column indices to each cell based on their geometric overlap relationships. Essentially, this groups the detected cells by rows and columns and resolves any spanning cells by checking which rows/columns they extend across. The end result is a reconstructed table: a complete structural description (often represented as an HTML or spreadsheet structure with each cell's coordinates in the grid).

An important aspect of TabGuard is that it achieves all of the above without using *Optical Character Recognition* (OCR) on the table content. Many traditional pipelines might first run OCR to find text bounding boxes and then infer structure from those. However, off-the-shelf OCR is unreliable on tables, often missing text in cells that have no gridlines or that span multiple lines. Instead, our content masking algorithm relies purely on image processing techniques that exploit the natural contrast between text and background. By using adaptive thresholding and contour detection, it robustly finds text regions even when OCR might fail (for example, when cell boundaries are present or text is rotated). This not only enhances privacy (since no actual text is ever decoded or seen, even on the client side) but also makes the masking extremely fast and resource-light. It can run in real time on commodity devices or mobile phones, preparing the image for secure processing without needing any GPU or cloud service. We show that our masking approach blacks out virtually *all* meaningful content in the table cells – covering 99.86% of the content area on benchmark datasets – significantly more than what even a high-quality OCR tool (we tested DocTR [108]) can localize (around 93.7% of token area). Thus, TabGuard's masking step provides strong privacy guarantees while retaining the essential structural cues (the positions of those masked regions) needed for TSR.

Overall, TabGuard takes the first steps toward **privacy-preserving table structure recognition**. Its combination of client-side content masking, grid-guided dynamic anchor generation, and structure-aware cell detection enables high accuracy without exposing sensitive content. We demonstrate through extensive experiments that TabGuard matches or exceeds state-of-the-art performance on several public benchmarks, even while operating on fully masked images. Notably, it effectively handles longstanding challenges such as very dense tables and complex spanning structures – scenarios where previous methods often struggled – all while ensuring that no private data is revealed to the recognition model. We make our code publicly available for the community to build upon this approach.

Contributions: In summary, the contributions of this work are as follows:

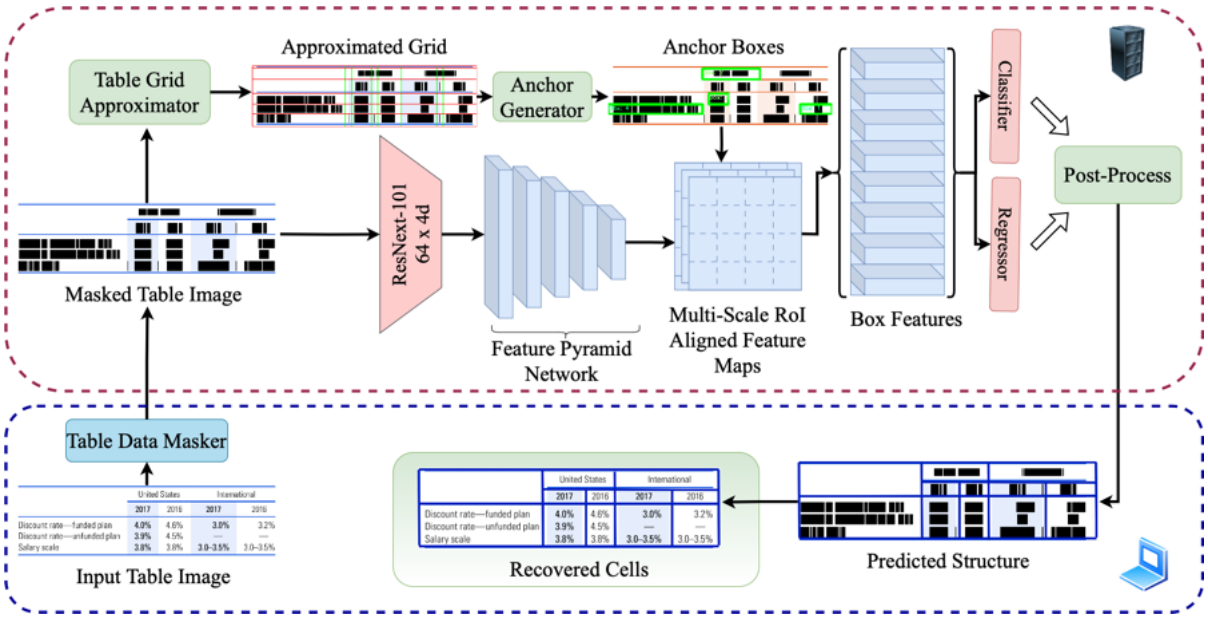


Figure 5.2: Architecture of *TabGuard*. The end-to-end pipeline consists of client-side operations (bottom) and server-side operations (top). On the client side, the original table image I_t is processed by the content masking algorithm to produce a masked image I_{mt} with all text rendered as filled boxes. Detected text contour coordinates are also sent (as JSON) to the server. The server then applies the Table Grid Approximator (TGA) to estimate the table's grid structure (dotted lines) from I_{mt} and generate dynamic anchors covering possible cell spans. These anchors are fed into the Table Cell Crypt Network (TCCN), which predicts the precise cell bounding boxes (green) without any OCR. A post-processing module then assigns row and column indices to each cell and refines boundaries. The final structured table (e.g., as HTML or a JSON of cells with their coordinates) is returned to the client. The client can optionally map the original text into each cell using the predicted cell coordinates, yielding a fully reconstructed table with content.

- We propose *TabGuard*, the first end-to-end TSR framework designed for *privacy preservation*. In TabGuard, all table content is masked on the client side, and only structure is processed on the server. This ensures that sensitive information remains secure, enabling practical deployment of TSR in domains requiring strict data privacy. Despite operating on masked data, TabGuard achieves accuracy on par with or better than state-of-the-art methods on standard benchmarks, effectively addressing challenges like high cell density and extreme aspect ratios while maintaining cross-domain robustness.

- We develop a fast, resource-efficient, and OCR-free **content masking algorithm** to remove all textual content from the table image. The proposed masking technique is language-agnostic and makes no assumptions about font or table style, allowing it to generalize to a wide variety of documents. By exploiting image processing (adaptive thresholding, line removal, and contour detection), it reliably blacks out text regions in any table. This step not only safeguards privacy but also standardizes the input for structure analysis.

- We introduce the **Table Cell Crypt Network (TCCN)** for detecting table cells on masked images, in conjunction with a novel two-stage strategy for structure recognition. In lieu of a conventional anchor generator and RPN, TCCN uses dynamic, table-specific anchors produced by our **Table Grid Approximator (TGA)**. TGA approximates the table's grid structure from the spatial distribution of text contours, and generates anchors that cover all possible cell spans. By integrating alignment and continuity losses into TCCN's training, we enforce that predicted cells form a well-aligned and continuous table structure. This leads to precise cell detection and makes the subsequent structure reconstruction (assigning row/column indices) straightforward via a simple post-processing step.

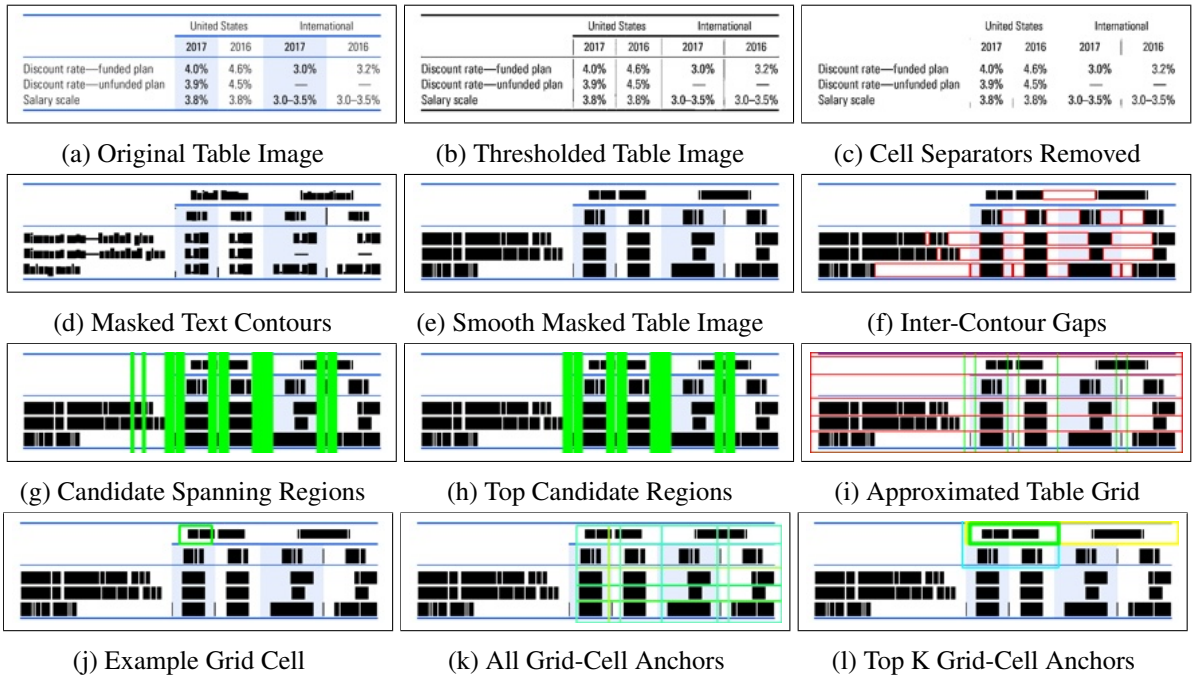
In the rest of this chapter, we detail the design of TabGuard and its evaluation. We describe the components of the TabGuard framework, including the content masking algorithm, the table grid approximation and dynamic anchor generation, the TCCN architecture, the specialized loss functions, and the post-processing for structure reconstruction. We present experimental results across multiple datasets, comparing TabGuard to existing methods and providing ablation studies that elucidate the impact of each component (masking strategy, TGA anchors, loss functions, etc.). We also discuss TabGuard's performance on challenging cases like dense or complex tables, and analyze the method's robustness under a privacy-preserving scenario. We conclude the chapter with a summary of our findings and their significance.

5.2 TabGuard: Privacy-Preserving TSR Framework

In this section, we provide an in-depth description of TabGuard's methodology. Figure 5.2 illustrates the overall architecture of the system. At a high level, TabGuard consists of several sequential components: (1) a client-side **Content Masking** algorithm that strips the table image of all textual data, (2) a server-side **Table Grid Approximator (TGA)** that analyzes the masked image to infer an approximate table grid structure, (3) a procedure for **Dynamic Anchor Generation** that uses the approximated grid to propose candidate cell regions, (4) the **Table Cell Crypt Network (TCCN)**, a detection model that identifies and localizes table cells among those candidates, and (5) a **Post-Processing** module that refines the cell geometry and reconstructs the table structure (assigning each detected cell a row and column index in the final table). Throughout the process, no actual cell content is utilized by the server—only the positional information of where content was. We next explain each of these components in detail, highlighting how they contribute to accurate structure recognition while preserving privacy.

5.2.1 Content Masking Algorithm

The first step in TabGuard is to remove all textual content from the table image on the client side, producing a masked image that reveals only structural cues. We design a **content masking algorithm** to accomplish this in a general, language-agnostic way. The masking procedure operates on the input table image I_t and outputs a masked table image I_{mt} of the same size, in which every character or text



(a) Original Table Image (b) Thresholded Table Image (c) Cell Separators Removed

(d) Masked Text Contours (e) Smooth Masked Table Image (f) Inter-Contour Gaps

(g) Candidate Spanning Regions (h) Top Candidate Regions (i) Approximated Table Grid

(j) Example Grid Cell (k) All Grid-Cell Anchors (l) Top K Grid-Cell Anchors

Figure 5.3: Steps (a) through (e) visualize the table's content masking algorithm. Steps (f) through (i) visualize the steps of table grid approximation. (j) through (l) show the anchor generation process for an example grid-cell.

region in I_t is covered by a solid opaque rectangle (effectively "blacking out" the text). Figure 5.3 (sub-figures (a)–(e)) provides a visual illustration of the stages of this algorithm, and Algorithm 3 formalizes the steps. Unlike naive methods that might require an OCR system to locate text (which could miss content or be language-dependent), our approach relies purely on image processing, making it robust across different scripts and layouts.

At a high level, the masking algorithm identifies regions of the image that contain text by exploiting the typically high contrast between text ink and the background paper. It then draws filled black boxes over those regions. The specific steps are as follows (refer to Algorithm 3):

After these steps, I_{mt} is a copy of the table where every piece of text has been covered by a black box, while non-text elements (lines, cell background, etc.) remain unchanged. The result of applying this content masking is shown in Figure 5.3(e): all table cell entries are obscured. The algorithm is fast and runs in linear time relative to the number of pixels ($O(H \times W)$ for an image of height H and width W), which makes it feasible on mobile devices or in web browsers without specialized hardware. Another advantage is that it does not depend on any external model or training – it works out-of-the-box for any table, regardless of language or font, since it purely leverages pixel intensity patterns.

Visualization of Masking Steps: Figure 5.3a–e correspond to key stages of the content masking process. Figure 5.3(a) shows an example original table image I_t. Figure 5.3(b) is the binarized image after

Algorithm 3 Algorithm to Mask Table Content.

Given a table image I_t, this algorithm generates a masked table image I_{mt} by masking all content with black rectangular contours.

1. Apply the popular **Projection Profile** algorithm [159] to estimate the skew (rotation angle) of the table. Rotate I_t by the negated estimated angle to correct any tilt, ensuring the table text lines are horizontally aligned.
2. Convert the de-skewed image to grayscale and apply **Gaussian adaptive thresholding** to produce a binary image. This separates text (foreground) from background by computing a threshold for each region based on local pixel intensities, effectively handling varying lighting or shading.
3. Remove all prominent horizontal and vertical lines in the binary image using a **Probabilistic Hough Line Transform**. This step detects lines (such as table ruling lines or borders) and eliminates them (e.g., by erasing those pixels) so that they do not interfere with text detection. By removing explicit table lines, we ensure that only textual regions remain as connected components.
4. Identify all connected components in the resulting binary image. Each connected component corresponds to a contiguous region of foreground pixels – in this context, these are candidate text regions (since we removed ruling lines, most remaining components should be text characters or words).
5. For each connected component, compute its bounding box (the smallest upright rectangle enclosing the component). These bounding boxes represent the "text contours," i.e. the approximate extents of text segments in the table.
6. Sort the list of text contour boxes primarily by their x-coordinate (left edge) and secondarily by their y-coordinate (top edge). This sorting ensures a consistent ordering (for example, left-to-right, top-to-bottom) which can be useful if needed for downstream processing or output, though the order does not affect the masking itself.
7. For each contour in the sorted list, draw a filled black rectangle on the original image I_t at the contour's location (i.e., replace the pixel region corresponding to that text with solid black). This produces the masked image I_{mt}.

adaptive thresholding; note how text regions appear as black blobs and the background is white. In Figure 5.3(c), the horizontal and vertical lines have been removed (one can see that ruling lines present in (b) are gone, leaving gaps where lines intersect text). Figure 5.3(d) visualizes the detected text contours (each text region's bounding box is highlighted). Finally, Figure 5.3(e) shows the masked image I_{mt} where those contours have been filled in black. This final masked image is what gets sent to the server.

The importance of this masking step is underlined by its effectiveness: we empirically found that our simple algorithm masks **99.86%** of the total content area in datasets like FinTabNet and SciTSR, whereas even a deep learning OCR method (DocTR [108]) covers only about 93.7% of the token area when extracting text boxes. In other words, our approach is extremely thorough at hiding content. By performing masking on the client, we ensure that the server sees no readable text, addressing the privacy requirement directly. Moreover, because the masking algorithm depends only on basic image operations, it is both *fast* and *lightweight*. Its runtime scales linearly with image size and it can run on CPUs in real-time; for example, a typical table image (e.g. 1024×1024 resolution) can be processed in a fraction of a second on a standard laptop or smartphone. This means the overhead introduced by privacy (masking) is minimal in practice. Finally, the algorithm is *language-agnostic* and *document-agnostic*: it does not matter whether the table text is English, Chinese, numeric, or symbols – as long as there is a contrast between text and background, the method will identify and mask it. Even stylized tables or unseen fonts are handled without issue, since no font-specific assumption is made.

After masking, the client transmits I_{mt} (and optionally the list of contour coordinates as auxiliary data) to the server. All subsequent processing occurs on the masked image I_{mt}. In summary, this content masking stage guarantees that TabGuard's structure recognition operates in a **privacy-preserving manner**: nothing but the positions of content (which themselves reveal minimal sensitive information) are available to the recognition model.

5.2.2 Table Grid Approximation (TGA)

Algorithm 4 Algorithm to Approximate Table Grid.

Given a masked table image I_{mt} of height H and width W, this algorithm outputs an approximate structural grid with n_r rows and n_c columns in an unsupervised manner.

1. **Identify candidate rows:** Group text contours by their vertical overlap to form *lines* of text. Each line represents content that lies on the same horizontal band (i.e. a potential table row). For each such line, examine the horizontal gaps between consecutive text contours in that line. Collect all these inter-contour gap segments across the line; denote the set of all gap segments as $empty_regions$.
2. **Generate column separator candidates:** Take each horizontal gap segment in $empty_regions$ and extend it vertically from the top of the table to the bottom (or as far as it can go without intersecting text). We score each such extended region based on two criteria: its width (thicker gaps are stronger evidence of a column break) and the percentage of table lines (rows identified in step 1) in which this region has no text intersection. Formally, for an extended gap region r, let s_w be its width and s_ℓ be the fraction of text lines that r spans without hitting a text box. We normalize both scores across all regions (to zero mean and unit variance, then scale [0,1]). These extended gap regions are our *candidate spanning regions* for column separators.
3. **Filter unlikely separators:** Discard any candidate spanning region that intersects text in a large fraction of lines. Specifically, if a candidate region passes through text in more than half of the total lines, it is likely cutting through a multi-column cell or is not a true separator, so we remove it.
4. **Rank and select separators:** Compute an aggregated score for each remaining region, $s = \sqrt{s_w^2 + s_\ell^2}$ (the Euclidean norm of the normalized width and line-span scores). This gives higher scores to regions that are both wide and span many lines without text (ideal separators). We then apply k-means clustering on these scores to split candidates into two groups (roughly "high score" vs "low score"). We select the cluster with the higher mean score as representing true separators. From that cluster, take the top 50 regions (at most) sorted by score as our initial set of column separators. The cap of 50 is empirically chosen to accommodate even extremely large tables (we assume no table will have more than 50 columns in practice).
5. **Ensure minimum grid granularity:** If the number of selected column separators is less than 20 (meaning the table might be very small or we were very conservative), we supplement by taking additional regions from the other cluster (the lower-score one) until we have 20 in total. This step ensures a baseline granularity – even if the table truly has fewer than 20 columns, having more separators than necessary is safer (they might result in some extra grid lines which can later be merged) than missing separators (which could cause multiple real columns to be erroneously merged). In summary, we impose a minimum of 20 vertical divisions for robustness.
6. **Construct the grid:** Using the horizontal lines (from step 1) as candidate row separators and the selected vertical regions (from steps 2–5) as column separators, assemble an approximate grid. Essentially, the table image is now divided by n_r horizontal lines and n_c vertical lines, where n_r is the number of text lines identified (which corresponds to the maximum row index of text) and n_c is the number of selected column divider candidates (after filtering and augmentation). This yields a grid of $(n_r \times n_c)$ cells covering the entire table area. Some of these grid cells correspond to actual table cells (or parts of them), while others may be spurious splits.

Given the masked table image I_{mt} (with text removed), TabGuard's next objective is to infer the basic grid structure of the table in an unsupervised manner. This step is crucial because it provides a structured prior for where rows and columns are likely to be, which we later use to generate anchors for

cell detection. We introduce the **Table Grid Approximator (TGA)** algorithm for this purpose. TGA analyzes the spatial distribution of the text contours (now effectively represented by the black boxes in I_{mt}) to deduce where the table's row and column separators could be. The output of TGA is an approximate grid: a set of horizontal lines (row boundaries) and vertical lines (column boundaries) that partition the table image into a grid of cells. Importantly, this approximated grid might be a finer partition than the actual table (i.e., it may "over-segment" some cells), but it provides a near-complete basis such that every true table cell corresponds to a union of one or more grid cells in this approximation.

Algorithm 5 Algorithm to Generate Candidate Anchors

Given a masked table image I_{mt} and its approximated grid (from TGA), this algorithm outputs candidate anchor boxes for TCCN in an unsupervised manner.

1. For each grid cell g_{ij} in the approximated grid, consider merges with up to 50 adjacent grid cells horizontally and 20 vertically. That is, generate anchors corresponding to every rectangular region that starts at g_{ij} and spans h columns and v rows, where $1 \leq h \leq 50$ and $1 \leq v \leq 20$, as long as the region stays within table bounds. This yields up to $50 \times 20 = 1000$ possible anchors for g_{ij}.
2. Among the generated anchors for cell g_{ij}, ensure that at least one anchor spans the entire image width (covers all columns) and at least 20 rows (or all remaining rows till bottom) to account for maximal spanning scenarios. (This is usually satisfied by the $h = 50, v = 20$ case or truncated to table size.)
3. Extract a set of geometric and positional features for each anchor candidate:
   - Number of text contours inside the anchor (ideally, a good anchor for a true cell should contain at least one text contour but not cut through any).
   - Number of *empty lines* the anchor spans (a line with no text contours within this anchor).
   - Total vertical gap (sum of vertical empty spaces) within the anchor.
   - The span of the anchor in terms of grid rows (v) and grid columns (h) it covers.
   - Indicators for whether the anchor touches the table boundary at top or bottom (presence of an empty line above or below the anchor, which might signify it's at the table edge or between cell blocks).
   - The aspect ratio and area of the anchor box.
4. Normalize all feature values to have zero mean and unit variance across all anchors, then scale them to [0,1] range. Using a labeled training set of tables, train a linear regression model (without bias term) to predict the Intersection-over-Union (IoU) of each anchor with the nearest ground-truth cell. The regression's weights thus learn which features make an anchor likely to overlap a true cell (higher IoU). We exclude a bias term so that the regression effectively computes a weighted sum of features that can be interpreted as a score.
5. At runtime (for an unseen table), compute the feature vector for each generated anchor and use the learned linear model to compute a **score** for that anchor (which is an approximate predicted IoU with a true cell). For each grid cell's set of anchors, select the top K highest-scoring anchors (in TabGuard we use $K = 10$). Discard all other anchors. The union of the retained anchors for all grid cells constitutes the final set of candidate anchors for TCCN.

The intuition behind TGA is that in a table, text is typically arranged in rows and columns. Even if the table has no drawn lines, the alignment of text naturally creates implicit rows (texts that are horizontally aligned across a table) and implicit columns (vertical alignment or consistent gaps between text in consecutive rows). By identifying these alignments and gaps, we can guess where the table's ruling lines would be if it were drawn as a grid. For example, consider two pieces of text one above the other in adjacent rows: if there is a noticeable vertical gap between them (with no text spanning that gap), that gap likely indicates a column boundary between two columns (since text from the two rows do not overlap horizontally, they must belong to different columns). Similarly, if there is a long horizontal gap on a

line (row) with no text, that might indicate a large empty cell or separation between two blocks of text, hinting at a column divider. TGA proceeds in several steps (Algorithm 4 provides a pseudocode outline).

The output of Algorithm 4 is an approximated grid layout for the table. Figure 5.3(f)–(j) depicts this process on an example. In Figure 5.3(f), we start with the masked image (with maybe slight smoothing applied to ignore tiny gaps). In Figure 5.3(g), every gap between text contours on each line is marked (yellow highlight for vertical gaps in one example row). In (h), these gaps have been stretched vertically into candidate spanning regions (blue vertical bands indicating potential column cuts). In (i), after scoring and clustering, the top candidates are selected – the blue lines indicate the final chosen column separator positions. And in (j), we see the resulting grid over the table: the image is partitioned into many small rectangular regions by the inferred row and column lines.

A few implementation details and optimizations are noteworthy:

- We treat the detection of overlapping text in step 1 efficiently by using an **Interval Tree** data structure. Each text contour can be represented as an interval on the x-axis (its horizontal span). To group contours into lines (rows), we need to quickly find, for each contour, which other contours overlap it vertically (i.e., lie in a similar y range) and then check horizontal gaps. The interval tree helps query overlapping intervals in $O(n \log n)$ time for n contours, speeding up the line grouping process.

- The parameters such as "50 maximum separators" and "minimum 20 separators" are chosen to balance over-segmentation vs. under-segmentation. The idea is that it's acceptable to have extra grid lines that do not correspond to real boundaries (we will handle them in anchor generation and detection), but missing a real boundary at this stage could merge two distinct columns into one region, which might then cause our anchor proposals to never consider that separation. By ensuring a sufficiently fine grid, we make sure to capture all true boundaries (at the cost of possibly including some false separators). Empirically, for large complex tables, this strategy works well: even if we include some false separators initially, the subsequent steps (dynamic anchors and the neural network) can learn to ignore or merge them, whereas if a true separator were missing, it would be much harder to recover.

- The complexity of TGA is $O(n \log n)$ for n text contours (due to sorting and the interval tree operations). In practice, n (the number of text boxes) is equal to the total number of text elements in the table, which is proportional to the number of characters or words. This can range from a few dozen to a few thousand in dense tables, but even for large n, the procedure is quite fast (negligible compared to neural network inference).

At the end of this stage, we have an approximated grid defined by sets of horizontal and vertical divider lines. Note that this grid is intentionally **over-complete**: it might have more divisions than

the actual table. For instance, if a table cell spans two columns, TGA might still draw a vertical line between those two columns (because from the text distribution alone, it looks like a gap). Similarly, a multi-row cell might be split by a horizontal line in the approximate grid. This is expected – TGA is not trying to perfectly recover spanning cells; rather, it is giving a framework that covers all cells. The over-segmentation will be corrected later by merging these grid cells when generating anchors and by the model's predictions. The crucial point is that after TGA, every actual table cell corresponds to one or more adjacent grid cells in the approximate grid. In other words, the TGA grid cells form a superset of the true table cells. This property guarantees that the next step (dynamic anchor generation) can consider combinations of grid cells to recover every true cell.

5.2.3 Dynamic Anchor Generation

The Table Grid Approximator yields a coarse grid that partitions the table into small cells. The next challenge is to use this grid to generate **anchor boxes** that are likely to tightly bound each actual table cell. We call this step *dynamic anchor generation* because the anchors are created on-the-fly for each table, derived from its content distribution, rather than being fixed across all tables. By tailoring anchors to each table's grid, we ensure high recall of true cells and help the detector focus on relevant regions.

Merging Grid Cells into Anchors: Consider the approximated grid from TGA. In that grid, a true table cell that spans multiple rows and/or columns will occupy a rectangular block of contiguous grid cells. If we could generate an anchor exactly covering that block, it would perfectly match the true cell. Therefore, the idea is to take each grid cell and simulate all possible rectangular merges with its neighboring grid cells up to a certain limit. Specifically, suppose the table's approximate grid has R rows and C columns of grid cells (so $R \times C$ grid cells in total). For each grid cell g_{ij} at row i, column j of this grid, we generate anchors corresponding to merging that cell with every combination of extending up to X columns to the right and Y rows down. In TabGuard, we set $X = 50$ and $Y = 20$ as the maximum column-span and row-span for anchors. These values are chosen based on observing that real tables seldom have a cell spanning more than 50 columns or 20 rows. (Even 50 columns is extremely large; typical spreadsheets might have tens of columns at most, but we allow up to 50 to be safe. Likewise, 20 rows span is generous for multi-row merged cells.)

By "recursively merge" we mean that for each grid cell we conceptually slide a window that can cover 1 to Y rows and 1 to X columns starting at that cell, and generate an anchor for each such window. The number of possible anchors from one base grid cell is $X \times Y$ in the worst case (if the table has at least X columns to the right and Y rows below that cell). With $X = 50$ and $Y = 20$, that is up to 1000 anchors per grid cell. For a table that has, say, 2000 grid cells (which could correspond to a moderately large table), this naive approach could yield up to $2{,}000 \times 1000 = 2{,}000{,}000$ anchors. Clearly, handling 2 million regions per image would be computationally infeasible directly. However, we will introduce a filtering mechanism to drastically reduce this number while retaining the relevant

anchors.

Before filtering, we also incorporate a couple of safeguards in the anchor generation:

- We ensure that for each grid cell, at least one of the generated anchors spans the entire width of the table and at least 20 rows vertically. This is to cover extreme cases where a cell might conceivably span to the table's full width or a very large number of rows (though such cases are rare). By guaranteeing these extreme anchors exist, we make sure not to accidentally miss a large spanning cell.
- We consider merges only in contiguous blocks to the right and downwards (to avoid duplicates). Since we generate from each grid cell, we will anyway cover merges in all directions if they exist. For instance, a cell spanning 2 rows up and 3 columns left from a given grid position will be generated when considering the top-left constituent grid cell of that spanning region.

Algorithm 5 outlines the anchor generation process in steps, and Figure 5.3(k)–(m) gives a visual example for a single grid cell. In Figure 5.3(k), we highlight one grid cell of the approximate grid. Figure 5.3(l) shows (a subset of) the many anchors generated by merging this cell with neighbors – you can see anchors of various sizes covering different spans. Figure 5.3(m) then shows the top few (top-K) anchors that are retained after scoring and filtering (in this illustration, K might be 5 for clarity). The best anchors tend to be those that align well with actual cell boundaries (in a real scenario, the ground truth cell covering that region would heavily influence the score).

Anchor Scoring and Filtering: The above generation produces a very large pool of anchors. We introduce a learning-based filtering mechanism to reduce this to a manageable number (on the order of a few thousand per image). The insight is that not all anchors are equally useful: many will have poor overlap with any true cell (for example, anchors that partially cut through a text box or span multiple cells). We want to keep anchors that are "good" proposals – those that likely correspond closely to a real cell. We achieve this by training a simple linear regressor on a set of geometrical features to predict an anchor's IoU with the nearest ground-truth cell. The features we use encode intuitive heuristics:

1. If an anchor intersects a text contour, that's a bad sign (anchors should ideally neatly contain text, not slice through it).
2. If an anchor spans many empty lines, it might be too large (covering multiple rows without content in between could indicate it's merging cells).
3. If an anchor spans many grid columns or rows (h or v large), it might be covering multiple cells unless those spans truly belong to one cell.
4. If the anchor has an empty line immediately above or below it, it could indicate it sits snugly between content, which is common for a proper cell bounding box.

5. The area and aspect ratio give a sense of the anchor's shape; extremely elongated anchors might be less likely to be a single cell unless text contours align accordingly.

(a) Good Candidate Anchors (b) Bad Candidate Anchors

Figure 5.4: Sample good and bad anchors. Good anchors have high overlap with ground-truth cells, contrary to bad anchors, which may have intersections with text regions.

We normalize these features and train a linear model (no bias) so that the output is simply a weighted sum of normalized features. This linear model is trained on a small set of ground-truth tables where we can compute IoU of each candidate anchor with actual cells, and find weights that give higher scores to anchors with higher IoUs. Essentially, it learns a scoring function that ranks anchor proposals.

At inference time, we compute this score for each anchor. Then, for each grid cell, we take only the top K anchors (by score) and discard the rest. We used $K = 10$ in our final system (earlier experiments used $K = 5$, but we increased to 10 to be safe; references in the text mention top-5 and top-10 at different places—our final implementation uses 10). By doing this, the vast majority of the millions of generated anchors are eliminated. The anchors that remain tend to tightly cover each text contour or small group of contours from various plausible spans.

After filtering, the total number of anchors is dramatically reduced. For example, if there are at most 2000 grid cells and we keep 10 per cell, we have at most 20,000 anchors. In practice, many grid cells (especially those corresponding to empty regions) might not even generate K distinct anchors that get high scores, so the final count is often even less (empirically a few thousand). This is a huge reduction from the initial combinatorial explosion, making it feasible to input these anchors into a detection network.

We emphasize that by generating anchors spanning every combination of up to 50 columns and 20 rows, we achieve extremely high **recall**: across FinTabNet and SciTSR test sets, **99.92%** of all ground-truth table cells have an IoU of at least 0.75 with *some* anchor in our proposed set. In other words, our anchor selection covers virtually all true cells, and with fairly tight overlap (0.75 IoU is a high threshold). This means TCCN will almost always have a good anchor candidate to focus on for each actual cell, which is critical for detection performance.

The complexity of anchor generation is manageable: generating anchors is done per grid cell (which we have at most a few thousand), and scoring them is linear in the number of anchors. The heavy lifting

of evaluating them further is left to the neural network, which will process only the retained anchors. The overall process up to now (masking, TGA, anchor generation) is completely unsupervised or uses very lightweight supervision (the linear regressor training) and happens quickly relative to the deep network inference.

Finally, Figure 5.4 provides a qualitative sense of what constitutes good vs. bad anchors. Good anchors (Figure 5.4(a)) typically align well with the true cell boundaries and encapsulate the text without cutting through it. Bad anchors (Figure 5.4(b)) might be shifted or have boundaries that intersect text or combine multiple text regions incorrectly. Our scoring mechanism aims to give higher scores to anchors of the first kind.

With the final set of candidate anchors now determined, we proceed to the detection stage. The anchors collectively cover all plausible cell locations in the table image, with a greatly reduced number for efficient processing. At this point, the TabGuard pipeline transitions from unsupervised heuristic processing to a trained model that will classify and refine these anchors.

5.2.4 Table Cell Crypt Network (TCCN) Architecture

Once we have a set of candidate anchor boxes likely corresponding to table cells, the next step is to actually detect the table cells among these candidates. We introduce the **Table Cell Crypt Network (TCCN)** as the detection model specialized for this task. The name "Crypt" alludes to working on encrypted (masked) content. In essence, TCCN is a modified Faster R-CNN detector that has been streamlined and optimized for table cell detection given our pre-computed anchors.

Backbone and Feature Extraction: TCCN uses a deep convolutional neural network backbone to extract features from the masked table image I_{mt}. We use a ResNeXt-101 64×4d model [158] as the backbone, which provides a strong feature representation. The backbone is typically augmented with a Feature Pyramid Network (FPN) to combine low-level and high-level feature maps, which is standard in modern object detectors to handle multi-scale objects. The output is a set of feature maps at various scales. Since the input is a single image containing a table (usually we ensure the table fits in the image by cropping or resizing as needed), we set a fixed input resolution (we resize all images to 1024×1024 in our implementation) to have a consistent scale for the network.

Region of Interest Align and Proposal Classification: In a standard Faster R-CNN, the next stage would be a Region Proposal Network (RPN) that proposes regions of interest. However, in TCCN we *do not need an RPN at all* because our dynamic anchor generation (TGA + merging + filtering) has effectively taken over the role of proposing regions. We already have a curated list of proposals (anchors) that we believe cover all table cells. Therefore, we eliminate the RPN and directly feed our anchors into the RoI Align and classification/regression heads. This significantly simplifies the network and training:

there is no need to tune RPN anchors or deal with NMS at the proposal stage. The "proposal" set is fixed by our algorithm per image.

For each anchor box, we perform **Multi-Scale RoI Align** on the FPN feature maps. This means we take the region of the image corresponding to that anchor and extract features by pooling (interpolating) from the feature pyramid (choosing the appropriate level based on anchor size and maintaining alignment). The result is a fixed-size feature tensor for each anchor, typically 7×7 spatial size with a certain depth (following conventions from Fast R-CNN).

These features are then fed into two sibling output branches:

- A **classification head**, which outputs a probability (or confidence) of this anchor being a valid table cell or not. Since we are only interested in one class (table cell) vs background, this can be a single logit or a two-class softmax (cell vs not-cell). We use a standard cross-entropy loss for classification during training.
- A **regression head**, which outputs refined coordinates for the bounding box of the cell. Even though the anchor is already a good candidate, we allow the network to adjust the box slightly to better fit the cell boundaries. This is done via the usual bounding-box regression approach (4 values usually: $\Delta x, \Delta y, \Delta w, \Delta h$ adjustments). We use an L_1 loss (smooth L_1 typically) on these deltas, calculated only for anchors that are labeled as positive (i.e., have IoU above a threshold with a ground-truth cell during training).

Positive/Negative Sampling: During training, we determine which anchors are considered positive examples and which are negative for the classification task. We use an IoU threshold of 0.5 to label anchors: if an anchor has IoU ≥ 0.5 with any ground-truth cell, it is a positive sample; if it has IoU below 0.5 with all ground-truth cells, it is a negative sample. (Anchors that have moderate IoU like between 0.3–0.5 could be ignored or considered negatives; we use a single cutoff for simplicity.) We also enforce a balanced sampling of positives and negatives in each batch. Specifically, we aim for a 1:1 ratio of positive to negative anchors when constructing a mini-batch for training. This prevents the vast number of negative anchors from overwhelming the learning signal. In practice, because our anchor generation is high-recall, there will be many more anchors than actual cells, so we sample or clamp the number of negatives to equal the number of positives in a batch.

Integration of Alignment and Continuity Losses: One of the distinctive aspects of TCCN is the incorporation of **alignment** and **continuity** losses into the training objective, inspired by prior works on table structure recognition that introduced these losses [1], [2]. These losses explicitly penalize misalignment of predicted cells that should align and gaps between cells that should be contiguous, thus injecting structural knowledge of tables into the model's learning. We adapt these losses with a smooth

L1 formulation (instead of the L2 used in the original works [1]) to improve stability.

We define the alignment loss $\mathcal{L}align$ as the sum of four components: row start alignment ($\mathcal{L}RS$), row end alignment ($\mathcal{L}RE$), column start alignment ($\mathcal{L}CS$), and column end alignment ($\mathcal{L}_{CE}$). These correspond to enforcing that:

- All cells that begin a particular row (i.e., cells whose top is the top of that row) should have the same Y_{top} coordinate.
- All cells that end a particular row (bottom of cells that are at the bottom boundary of that row) should have the same Y_{bottom}.
- Similarly, all cells that start a given column should share the same X_{left}, and all that end a column share the same X_{right}
- All cells that start a given row should share the same Y_{top}, and all that end a row share the same Y_{bottom}

We collect these by grouping predicted cells by their row and column indices. However, since TCCN is predicting bounding boxes, it doesn't inherently produce row indices. We infer these groupings on the fly using an approximation: if two predicted cells significantly overlap in the vertical direction, they are likely in the same row (or one spanning multiple rows, but alignment loss mainly targets common boundaries like the top of all cells in a row segment). In practice, during training we can use ground-truth alignment (knowing which predicted cells correspond to the same row in truth) or encourage alignment with nearest neighbors. Our implementation followed the formulation in [1]: for each ground-truth row r, we take all predicted cells i, j that belong to that row and add a term $|Y^i_{top} - Y^j_{top}|$ for L_{RS} (and similarly for Y_{bottom} for L_{RE}). These are summed for all pairs in the row. We do analogous for columns (X_{left} and X_{right} differences for cells in the same column group) for $\mathcal{L}CS$ and $\mathcal{L}CE$. All these differences are taken absolute (or L1 norm). We then sum them up for a total alignment loss $\mathcal{L}_{align}$. In practice, we apply smooth L1 to each difference to reduce sensitivity to outliers.

The continuity loss $\mathcal{L}_{cont}$ targets the gaps between consecutive cells. Specifically, for any two cells that are vertically adjacent in the same column (meaning one's start row is exactly one more than the other's end row), we penalize the vertical gap between them. This encourages that if one cell ends and the next begins immediately below it (with no intermediate row), the bottom of the top cell should equal the top of the bottom cell (no gap). Similarly, for horizontally adjacent cells in the same row (one starts at the next column right after the other ends), we penalize the horizontal gap between them. By adding $\mathcal{L}align$ and $\mathcal{L}cont$ to the network's loss (with some weighting), we bias TCCN to predict cells that form a properly aligned grid with minimal gaps, essentially reconstructing the table's structure. These losses are only applied to predictions that correspond to real neighboring cells; if a prediction is a false positive or isolated, it might not contribute. In training, we weigh these losses with a small factor (we used 0.01

as regularization weight for each in our experiments) so that they supplement but do not dominate the standard detection losses.

$$\mathcal{L}_{RS} = \sum_{r \in R^{\text{start}}} \sum_{i,j \in \text{row } r} \left\| Y^i_{\text{top}} - Y^j_{\text{top}} \right\|_1,$$
$$\mathcal{L}_{RE} = \sum_{r \in R^{\text{end}}} \sum_{i,j \in \text{row } r} \left\| Y^i_{\text{bottom}} - Y^j_{\text{bottom}} \right\|_1,$$
$$\mathcal{L}_{CS} = \sum_{c \in C^{\text{start}}} \sum_{i,j \in \text{col } c} \left\| X^i_{\text{left}} - X^j_{\text{left}} \right\|_1,$$
$$\mathcal{L}_{CE} = \sum_{c \in C^{\text{end}}} \sum_{i,j \in \text{col } c} \left\| X^i_{\text{right}} - X^j_{\text{right}} \right\|_1.$$

$$\mathcal{L}_{align} = \mathcal{L}_{RS} + \mathcal{L}_{RE} + \mathcal{L}_{CS} + \mathcal{L}_{CE}$$

$$\mathcal{L}_r = \sum_{i,j \in \ cells} ||Y^i_{top} - Y^j_{bottom}||^1_1 \cdot I\big(R^i_{start} == R^j_{end} + 1\big),$$
$$\mathcal{L}_c = \sum_{i,j \in \ cells} ||X^i_{left} - X^j_{right}||^1_1 \cdot I\big(C^i_{start} == C^j_{end} + 1\big)$$

$$\mathcal{L}_{cont} \ = \ \mathcal{L}_r + \mathcal{L}_c$$

Eliminating the RPN and using pre-computed anchors has an additional benefit: it streamlines and stabilizes training. We do not have to alternate between RPN and detector training or worry about proposal recall. The proposals are essentially fixed and of high quality. As a result, TCCN can converge faster and is less prone to the sorts of missed detections that plague anchor-based detectors with poor initial anchors. The heavy ResNeXt-101 backbone ensures that even though the content is masked, the network can extract features like the edges of the masked regions, subtle column separators, etc., to accurately localize cell boundaries.

At inference time, TCCN takes the masked image and our anchor list, performs RoI Align, and outputs a set of detected cell boxes with confidence scores. Because the model learned to align and keep continuity, these boxes typically already form a coherent table structure (aligned rows/columns). We then apply the final step to translate these detections into the structured table output.

5.2.5 Post-Processing for Structure Recovery

The final stage of TabGuard is a post-processing step that converts the raw predictions of TCCN (which are essentially just a set of bounding boxes for cells) into an actual table structure with row and

column indices, while also refining the box coordinates for perfect alignment. This step is **dataset-agnostic** and rule-based, not learning-based, and it ensures the output is in a readily usable format (for example, a list of cells each annotated with which row and column it belongs to in the table, including spanning information).

Method	CAR-F1			Struct-TEDS		TEDS		AP_{50}	
	IC-13	*SciTSR*	*cTDaR*	*FTN*	*PTN*	*FTN*	*PTN*	*FTN*	*PTN*
GraphTSR [24]	87.2	95.3	-	-	-	-	-	-	-
SPLERGE [18]	95.0	92.6	-	-	-	-	-	-	-
LGPMA [67]	95.3	98.8	-	-	96.7	-	94.6	-	-
TSRFormer [69]	-	**99.6**	-	-	97.5	-	-	-	-
CascadeTabNet [160]	-	-	43.8	-	-	-	-	-	-
GTE [4]	93.5	-	45.9	91.0	93.0	-	-	-	-
TGRNet [56]	66.7	-	82.8	-	-	-	-	-	-
GuidedTSR-AO [161]	95.46	-	-	-	-	-	-	-	-
SEM [162]	-	-	-	-	-	-	93.7	-	-
EDD [163]	-	-	-	90.6	89.9	-	88.3	-	79.2
TableFormer [68]	-	-	-	96.8	96.8	-	93.6	-	82.1
MTL-TabNet [164]	-	-	-	**98.8**	97.9	-	-	-	96.7
TabStructNet [1]	90.6	92.0	58.3	-	-	-	90.1	-	-
VAST [149]	96.5	99.5	58.6	98.6	97.2	**98.2**	96.3	-	94.8
NCGM [70]	98.8	98.8	85.3	-	95.4	-	-	-	-
LORE [144]	98.9	98.7	88.3	-	**98.1**	-	-	-	-
GridFormer [150]	-	99.3	-	98.6	97.0	-	95.8	-	-
Faster RCNN* [30]	84.2	85.3	33.4	78.8	80.3	76.1	77.4	71.5	72.6
RetinaNet*	83.6	86.2	32.4	77.3	80.1	75.4	77.1	70.8	72.2
YOLO v9 †	89.6	90.2	40.3	83.6	86.1	81.8	83.5	77.4	78.7
Deformable-DETR* [147]	92.2	93.9	61.4	91.7	92.4	-	-	87.3	89.1
Anchor-DETR* [157]	95.4	96.8	70.2	94.9	95.6	-	-	91.0	92.3
RetinaNet †,*	98.6	99.0	85.3	96.8	97.2	95.1	95.6	93.6	93.8
TabGuard^{cell}	**99.2**	99.1	**89.9**	97.8	**98.1**	97.1	**97.3**	**95.7**	**96.2**
TabGuard^{content}	**99.2**	99.2	NA	98.1	**98.3**	97.1	**97.3**	**95.9**	**96.4**

Table 5.1: Comparison using CAR-F1 scores at IoU=0.5 on IC-13, SciTSR, cTDaR datasets; and using S-TEDS and TEDS on FTN and PTN datasets. Training and testing environments for each test dataset is consistent across methods for fairness. Method M* includes alignment and continuity losses [1], [2], and M† uses anchors from TGA. *TabGuard* has been trained and tested using masked table images. *TabGuard*^{cell} evaluates cell-level bounding boxes, and *TabGuard*^{content} evaluates content-level bounding boxes, ensuring fair comparison across different environments. Since *TabGuard* generates rectangular bounding boxes, it does effectively handle misaligned or curved tables. Therefore, we opt not to compare our method on the WTW dataset.

We perform the following in post-processing:

1. **Refine Cell Boundaries:** Although TCCN's predictions are generally well-aligned due to the losses, minor misalignments can occur (especially if the model slightly under/overshot a boundary or if two adjacent cells have a tiny gap/overlap). We correct these by examining the set of all

detected boxes. We snap the boundaries that are supposed to be shared to a common value. Concretely, we look at all predicted x-coordinates of left and right edges, and all y-coordinates of top and bottom edges, and cluster them (values that are within a few pixels are averaged and replaced by the average). This effectively forces nearly aligned edges to become exactly aligned. We also identify if any predicted cells overlap or leave small gaps: if two cells in the same row overlap slightly or have a tiny gap, we adjust their edges to either meet or not overlap, based on majority alignment. These refinements use simple heuristics, assuming that true tables have perfectly contiguous or aligned cell borders. The result is that the set of box coordinates now can be interpreted as a consistent grid.

2. **Assign Row and Column Indices:** We then assign each box an integer row index and column index. We do this by sorting the boxes by their coordinates. We sort all cells by their y (top) coordinate to determine row groupings. For each unique top coordinate (after refinement, top coordinates that are equal indicate the same row start), assign an increasing row number. Within each row grouping, we sort cells by x (left) coordinate to assign column indices. Each unique left coordinate becomes a new column. We also account for spanning: if a cell's height covers multiple row spans (i.e., its bottom coordinate aligns with the bottom of a lower row beyond the one its top started at), that cell is a multi-row spanning cell. Similarly for column span via widths. We determine $R_{start}, R_{end}, C_{start}, C_{end}$ for each cell box by seeing between which row lines and column lines it lies. Cells that span multiple rows or columns will naturally have fewer neighbor cells in those rows/columns (we might find empty "slots" under a spanning cell which indicate that region is covered by the spanning cell).

3. **Output Structured Format:** Using the assigned indices, we construct the table in a structured format. For example, an output could be a list of table cells where each cell has coordinates $(R_{start}, C_{start}, R_{end}, C_{end})$. This can be readily converted to HTML (using <td rowspan=.. colspan=..> to represent spanning) or other formats like CSV (with empty entries for spanned cells) or JSON.

Because the input to this process was a masked image, we currently have structure but no content for each cell. In a deployment of TabGuard, once the client receives this structured table (essentially the skeleton of the table), it can perform a final mapping of content. The client knows the original image I_t and the coordinates of each cell (now in the context of the original image's coordinate system, which we maintain consistent through resizing). For each cell bounding box, the client can either extract the text from the original document (if the original is a PDF, one could use PDF text extraction in that region), or run an OCR on that specific cell region on the original image. Since the content within each cell is relatively limited (and potentially OCR can be done locally cell by cell if needed), this is feasible. The key is this happens on the client side with the original content, meaning the server still never sees the actual text. The client then populates each cell's content in the structured output, yielding the final reconstructed table with both structure and content fully restored. The result is an end-to-end system

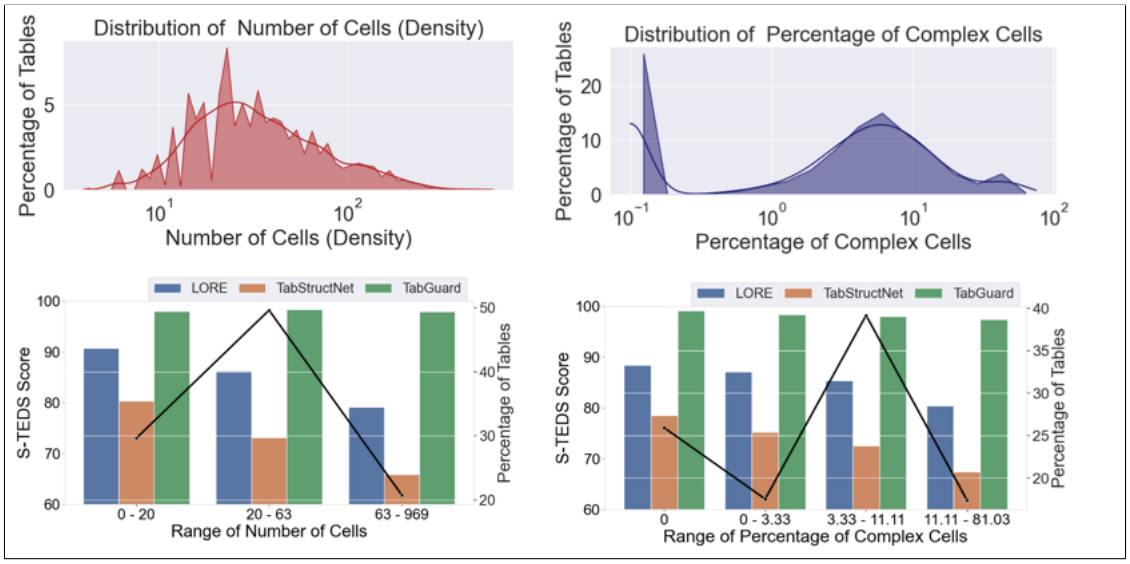


Figure 5.5: Top-Left and Top-Right plots indicate log-scale distribution of table density with respect to number of cells and tables with varying complexity (multi-column/row/line) of cells. Bottom-Left plot compares performances of LORE [144], TabStructNet [1] and *TabGuard* with against varying table densities. Bottom-Left plot compares performances of LORE [144], TabStructNet [1] and *TabGuard* with against varying table complexities. All distributions and performances are measured on FinTabNet-Test dataset with S-TEDS evaluation metric with masked table images. Black line in the second row shows the linear-scale dataset distribution.

where the only server input was I_{mt} (masked image) and the output from the server was a structured table (with cell coordinates). All actual sensitive text stays on the client side and is only merged at the end. This design satisfies the privacy-preserving objective while delivering a high-fidelity table reconstruction.

To summarize, the post-processing uses geometric overlap and alignment to finalize the table structure. Cell overlaps are the sole criterion used to infer row/column grouping: if two cells overlap in the horizontal span, they are in the same column group (or one spans into that column); if they overlap vertically, they share column or indicate spanning. This simple rule is surprisingly sufficient because our detection has enforced such strong alignment. Essentially, one can draw vertical lines at each unique detected x position and horizontal lines at each unique y position; then each cell's indices correspond to which intervals it spans. We deliberately keep this logic simple and not machine-learned, to avoid overfitting to any particular dataset's quirks. It works across datasets (as it encodes generic table structure principles).

After this step, TabGuard has successfully reconstructed the table's structure entirely on masked data. The predicted tables can be evaluated against ground truth (as we do in experiments) using metrics like structural accuracy or cell adjacency metrics, all without ever exposing the content. In the next

section, we will discuss the experimental evaluation of TabGuard, including dataset details, implementation specifics, and a comprehensive comparison with existing methods. We will also present ablation studies that highlight the contribution of each component (masking, TGA, anchors, losses) to the overall performance.

5.3 Experiments and Results Analysis

5.3.1 Implementation

We resize all images to a resolution of 1024×1024. Anchors having an IoU overlap of more than 0.6 with a ground truth box are used as positive and others as negative samples for training in a balanced manner. Our model can be trained on a single NVIDIA 1080TI GPU with a batch size of 2. Regularization parameters corresponding to alignment and continuity losses are set to 0.01. We smooth out the cell boxes by identifying overlaps along X and Y axes, and the final coordinates are translated to PDF coordinates to extract content.

Method/IoU	**0.6**	**0.7**	**0.8**	**0.9**	**W.Avg**
NLPR-PAL	0.37	0.31	0.20	0.04	0.21
CascadeTabNet	0.44	0.35	0.19	0.04	0.23
TabGuard	0.86	0.73	0.31	0.07	0.446

Table 5.2: Comparison on IC19 Track B2 on varying IoU thresholds.

5.3.2 Datasets and Evaluation

We use FinTabNet [4] (FTN) and SciTSR [24] datasets for training. We evaluate TabGuard on FinTabNet and PubTabNet [163] (PTN) datasets using Tree Edit Distance Similarity (TEDS) [163] and Structural TEDS (S-TEDS, ignoring cell content). For ICDAR-2013 [48] (IC-13), SciTSR, and ICDAR-2019 [41] (cTDaR) datasets we use F1 score on Cell Adjacency Relations (CAR-F1) for evaluation [48]. Since *TCCN* has alignment & continuity constraints, we split the horizontal & vertical gaps between every adjacent pair of cells equally and extend their boundaries to ensure proper alignment. We consistently use IoU of 0.5 for all evaluations. We evaluate TabGuard on both cell-level and content-level bounding boxes. To obtain content-level boxes, we identify masked contours within a predicted cell and accordingly identify the predicted content bounding boxes within each table cell.

5.3.3 Comparative Study

We assess the performance of our method on scanned and cropped table images and table images extracted from PDF documents. To ensure consistency in comparisons, we additionally present our results using content-level bounding boxes, as depicted in the last row of Table 5.1. To derive content-level boxes, we identify masked contours within a predicted cell and utilize these coordinates to determine

Method	Train Mask	Test Mask	S-TEDS
LORE [144]	✗	✗	96.7
LORE [144]	✗	✓	71.1
LORE [144]	✓	✓	85.2
TabStructNet [1]	✗	✗	89.8
TabStructNet [1]	✗	✓	64.5
TabStructNet [1]	✓	✓	72.8
TabGuard	✗	✗	93.6
TabGuard	✗	✓	91.4
TabGuard	✓	✓	98.2

Table 5.3: Presents a comparison of training and testing variations using masked and unmasked table images for *TabGuard* against LORE [144]. We maintain consistency by utilizing the FTN-Train and FTN-Test datasets for training and evaluation. When testing *TabGuard* on original images, OCR bounding boxes are employed for grid approximation and anchor generation.

each table cell's expected content-level bounding boxes. As indicated in Table 5.1, our method surpasses all prior approaches on the ICDAR-2013 and cTDaR datasets, achieving a margin of 0.3% and 1.3% Cell Adjacency Relation F1 scores, respectively. Regarding the ICDAR-2013 dataset, we follow a consistent evaluation protocol employed by [1], [70], [144], where a partial dataset was utilized for fine-tuning and evaluation respectively. In case of the cTDaR dataset, we compute the results using an IoU threshold of 0.5, following the approach of the competitive baseline method GTE. Table 5.1 presents the weighted average F1 score. On the SciTSR, our method surpasses the performance of most prior methods, achieving an F1 score within a 0.5% range of SoTA while maintaining content privacy. TabGuard achieves state-of-the-art S-TEDS and TEDS scores on PubTabNet. Nonetheless, we observe that some images in the FinTabNet dataset have incorrect annotations, particularly those containing multi-row spanning cells[1]. Interestingly, our model generates the correct structure for such cases compared to the original annotation. Additionally, we report the performance of cell detection using average precision (AP) at an IoU threshold of 0.5 on the PubTabNet and FinTabNet datasets. Table 5.2 compares the performance of TabGuard on ICDAR-2019 Track B2 dataset on varying IoU thresholds. It is crucial to highlight that input to our solution for all datasets are masked table images. Our findings show that the predominant characteristic in identifying the table layout is the location of content rather than the content itself.

5.3.4 Comparison in Privacy Preserving Scenario:

To evaluate our method against the current state-of-the-art, we fine-tune LORE [144] and TabStructNet [1] on masked table images from the FTN-train dataset using their respective open-source implementations. While it might seem intuitive that masking content should enhance the performance of all existing methods, we observe from Table 5.3 that the performance of [1], [67], [144] decreases no-

[1]Details of such instances are available in our supplementary material.

Method	*CAR-F1*			S-TEDS	
	IC-13	*Sci-C*	*IC-19*	*FTN*	*PTN*
Faster RCNN	81.3	81.7	31.1	76.5	79.4
Faster RCNN^{DP}	74.1	74.6	21.5	71.2	72.6
Faster RCNN^{OCR}	82.4	82.8	27.6	77.4	78.9
Faster RCNN^{CM}	84.2	85.3	33.4	78.8	80.3

Table 5.4: Impact of privacy strategy. DP, OCR and CM indicate additional, differential privacy, OCR content masking and contours based content masking. alignment and continuity losses are used for all. TGA and custom anchors are not used for fairness.

tably when trained and tested on masked images. The incorrect cases primarily arise in cells spanning multiple lines and where the inter-contour gap is more comprehensive than average within the same cell. We attribute the superior performance of *TabGuard* on masked images to the anchors generated using the *Table Grid Approximator*, which provides additional cues to the model to aid in table structure recognition.

5.3.5 Ablation Study

We perform a series of ablation experiments to validate the efficacy of our proposed modules. Table 5.4 compares use of differential privacy, OCR based content-masking and our contour based content masking for ensuring privacy preservation. For a fair ablation study, we use Faster RCNN as the fixed architecture augmented by alignment and continuity losses [1], [2] without using anchors from TGA. Table 5.5 illustrates that our model, trained on masked images, can proficiently process images with unmasked content during testing. For the unmasked regions, we utilize DocTR [108] OCR to acquire cell-level bounding boxes employed in grid approximation and anchor generation. The findings also indicate that performance remains consistent across training and testing domains, especially when the majority of the table's content is masked. Table 5.6 shows the effectiveness of our TGA and anchor generation for table cells detection. The reduced number of anchor boxes reduces the search space for the global optimum. It improves optimization performance by 2.5 times[2]. Prior approximation and generation of anchor boxes also allow for better performance in case of unseen table styles in a cross-domain setup. Table 5.6 highlights the fact that switching the network backbone from ResNext-101 to ResNet-18 leads to small impact on performance while significantly reducing the number of parameters.

5.3.6 Dense and Complex Tables

To study the effectiveness of our method on densely packed and complex (multi-row/multi-column and multi-line spanning cells) tables, we analyze the comprehensively analyze characteristic distributions and compare our method against LORE [144] and TabStructNet [1]. For fairness, we use open-source implementations of the two methods, fine-tune them using masked images from the FinTabNet-

[2]Qualitative examples and details on anchors distribution and optimization are in the supplementary material.

Train Dataset	Test Dataset	% Content Masked				
		100%	75%	50%	25%	0%
SciTSR	SciTSR	99.1	98.3	96.5	94.2	92.5
SciTSR	FTN	96.4	93.2	92.9	89.3	86.4
FTN	SciTSR	98.8	97.1	95.9	94.1	92.3
FTN	FTN	98.2	97.6	96.4	95.1	93.7

Table 5.5: Impact of content masking on domain adaptation. All quantitative scores are measured in terms of CAR-F1 scores.

BackBone	Anchors	#Model Params	CAR-F1 Score
ResNet -18	RPN	30.3M	88.3
	TGA	29.0M	97.1
ResNxet-50 32x4d	RPN	40.2M	89.7
	TGA	41.5M	97.5
ResNxet-101 64x4d	RPN	98.6M	91.1
	TGA	99.9M	98.2

Table 5.6: Illustrates the impact of inductive bias and backbones through training and testing on the FinTabNet dataset. CAR-F1 scores on Cell Detection at the IoU threshold of 0.5 are reported.

Train dataset, and evaluate them on the FinTabNet-Test dataset. Figure 5.5 demonstrates our method's superiority in privacy-preserving scenarios across varying table densities and complexities.

5.3.7 Impact and Limitations

TabGuard is specifically tailored for scanned images of cropped tables, focusing on extracting table structures in a controlled 2D environment. It does not extend to scenarios where tables are captured using camera devices, with challenges such as curvature, perspective distortions, or 3D effects. Moreover, our approach does not support tables with complex embedded entities, such as nested tables, graphs, or images, which may require more advanced parsing techniques.

However, in our experiments with skewed images—where tables appear slightly rotated, we found that applying skew correction as a preprocessing step successfully addressed minor misalignments, however it may not be sufficient for more severe distortions or images with significant perspective changes.

5.4 Conclusion

In summary, this chapter presented *TabGuard*, a novel privacy-preserving framework for table structure recognition that enables accurate table parsing without revealing sensitive content. The key technical contributions of TabGuard span several components. First, a client-side content masking algorithm irreversibly conceals all textual information in the document before any processing, ensuring that only

structurally-relevant cues (such as character bounding boxes or layouts) leave the client. Building on this secure representation, TabGuard introduces a *Table Grid Approximation (TGA)* module to infer a coarse grid structure of the table from the masked content. This approximated grid provides a strong inductive bias about row and column layouts, which in turn drives a dynamic anchor generation strategy for cell detection. Instead of relying on predetermined anchor sizes or aspect ratios, TabGuard generates table-specific anchors that span each detected text region, thereby effectively handling cells of widely varying size and aspect ratio. These anchors feed into a streamlined Table Cell Crypt Network (TCCN) detector – a convolutional neural network tailored for table cells – which we augment with novel alignment and continuity losses to enforce that detected cells align perfectly in rows and columns and maintain a continuous table grid. A lightweight post-processing step then uses the well-aligned cell detections to reconstruct the full table structure, by assigning appropriate row and column span indices to each cell. Together, these contributions constitute an end-to-end pipeline that robustly recovers table structures while rigorously safeguarding the underlying data.

A central outcome of TabGuard is its ability to reconcile high-performance table recognition with strict privacy requirements. By operating exclusively on masked data and performing all sensitive computations on the client side, TabGuard ensures that confidential information in documents remains protected at all times. This makes the approach particularly attractive for sensitive domains such as finance, healthcare, legal, and government, where documents often contain private or regulated information. Importantly, our extensive experiments demonstrate that this privacy-by-design does not come at the cost of accuracy. TabGuard achieves state-of-the-art structure recognition performance across diverse public benchmarks, matching or even surpassing traditional TSR models that operate on unmasked data. It shows strong generalization and robustness in handling challenging scenarios – from dense tables with tens of rows and columns to documents with complex layouts and multi-span cells – all while remaining agnostic to the actual content of the tables. These empirical results underscore that the preservation of privacy need not compromise model effectiveness: TabGuard manages to excel in structure recognition tasks even under the imposed constraint of content masking.

The advances presented in TabGuard mark an important step forward for secure document analysis. By achieving high accuracy in TSR under rigorous privacy constraints, TabGuard sets a precedent for developing privacy-preserving methods in other document understanding tasks. The principles and techniques introduced in this framework – such as content masking coupled with structure-preserving cues, and detection models guided by alignment continuity constraints – could be extended to related problems like form parsing, structured data extraction from sensitive records, or any scenario requiring insight from documents without exposing their contents. Future work may explore integrating TabGuard's approach with complementary privacy-enhancing technologies (for example, leveraging homomorphic encryption or secure enclaves for model inference, or adopting federated learning to train on distributed confidential datasets) to further harden the end-to-end security of the system. In essence,

the significance of TabGuard lies not only in addressing an immediate need for privacy-preserving table recognition, but also in laying down a foundation for next-generation document analysis solutions that are both high-performing and trustworthy. By bridging advanced table structure recognition with stringent privacy safeguards, TabGuard paves the way for broader adoption of AI-driven data extraction in privacy-critical applications, heralding a future where secure and effective document understanding go hand in hand.

Chapter 6

EviFiVQA: Evidence Grounded VQA on Financial Table Images

6.1 Introduction

Financial statements serve as the primary foundation for evaluating corporate performance, detecting fraud, and meeting regulatory mandates [165]–[167]. Consequently, analysts, auditors, and regulators rely heavily on these documents to form high-stakes judgments. In such contexts, evidence traceability becomes indispensable: any conclusion drawn from a financial report must be backed by explicit references to the original tables. Without this level of evidence localization, AI-driven insights risk being opaque and can compromise the reliability of the decision-making process. However, financial documents are often lengthy and complex, making manual verification of every reported figure impractical. An automated system for financial analysis must not only compute answers to quantitative questions correctly but also point to the exact locations in the statements that justify those answers, to ensure auditability and transparency in its reasoning.

Due to the heterogeneous and often complex nature of financial statements, which commonly feature multi-year column layouts, merged cells, footnotes, and disclaimers, traditional Optical Character Recognition (OCR) pipelines may introduce digit misreads and break the inherent link between the numeric content and the document's visual-spatial structure [1], [144]. These issues create difficulties in validating which cells support a given conclusion and complicate subsequent audits. For example, if an OCR-based system extracts financial table data into text form, it might lose the context of which row or column a number came from, especially in tables with nested structures or spanning cells. Verifying the origin of a particular figure then becomes cumbersome, undermining trust in the system's outputs. Moreover, minor OCR errors (e.g., misrecognizing a digit) can propagate through calculations and lead to significantly incorrect answers without an obvious way to trace the mistake back to a specific cell in the document.

As an alternative, large Vision-Language Models (VLMs) can directly analyze document images, reducing the vulnerability of text-based pipelines to OCR errors and maintaining a clearer visual-spatial reference [58]–[60], [105], [168]. By combining layout analysis with textual understanding, these mod-

els can reconcile data inconsistencies in annual reports, detect misrepresentations in profit-and-loss statements, and highlight disclaimers that affect numeric interpretations. For instance, a VLM can observe that a footnote symbol is associated with certain values and include that context in its reasoning. When paired with external accounting knowledge, such models can also identify anomalies in key financial metrics, providing a robust foundation for comprehensive risk assessments. Importantly, by generating bounding-box evidence on the source documents, these models can facilitate transparency, auditability, and interpretability. Finance professionals or regulatory bodies can directly inspect the exact segments of financial statements that inform each inference—a critical feature for downstream applications such as portfolio stress testing, fraud detection, credit risk modeling, and automated compliance checks. In other words, the model's answer is not just a number or text, but accompanied by highlighted cell locations in the document, indicating "these are the cells I used to arrive at my answer." This capability greatly increases trust, as users can verify the answer against the original document.

However, because most large-scale VLMs are trained on general-purpose datasets, their specialized performance in financial document analysis remains uncertain. General VQA models and even sophisticated large language models have not been rigorously tested on tasks requiring multi-step numerical reasoning over complex tables or on producing audit-ready outputs. Addressing this gap necessitates a structured benchmark to systematically evaluate and refine VLMs for financial tasks. To this end, we introduce EviFiVQA (`https://github.com/sachinraja13/EviFiVQA`) – a large-scale evidence-based financial Visual Question Answering dataset built on real-world financial statements. EviFiVQA is designed to push the boundaries of reliability and transparency in AI-powered financial analytics by requiring models to produce not only correct answers to financial questions but also explicit evidence for those answers in the form of cell bounding boxes on the document images. In doing so, our benchmark forces models to perform interpretable, multi-hop reasoning that mimics the work of a human financial analyst gathering evidence for an answer.

EviFiVQA focuses on challenging scenarios encountered in financial analysis, including multi-step numeric computations (such as year-over-year growth calculations or ratio analysis across rows and columns), hierarchical table structures (which require understanding subtotals and parent-child relationships in financial reports), and the necessity of explicit bounding-box localization for each supporting cell in a table. Figure 6.1 illustrates an example where a state-of-the-art language model (ChatGPT-4 with vision capabilities) computes the correct numeric answer but fails to select the right evidence cells: the model's highlighted cells (red) do not align with the true supporting cells (green). This discrepancy underscores how a model might "get the right answer for the wrong reasons"—unacceptable in high-stakes finance settings where justification is as important as correctness. By requiring evidence grounding, EviFiVQA penalizes such behavior and directs model development toward methods that truly understand and use the content of financial tables.

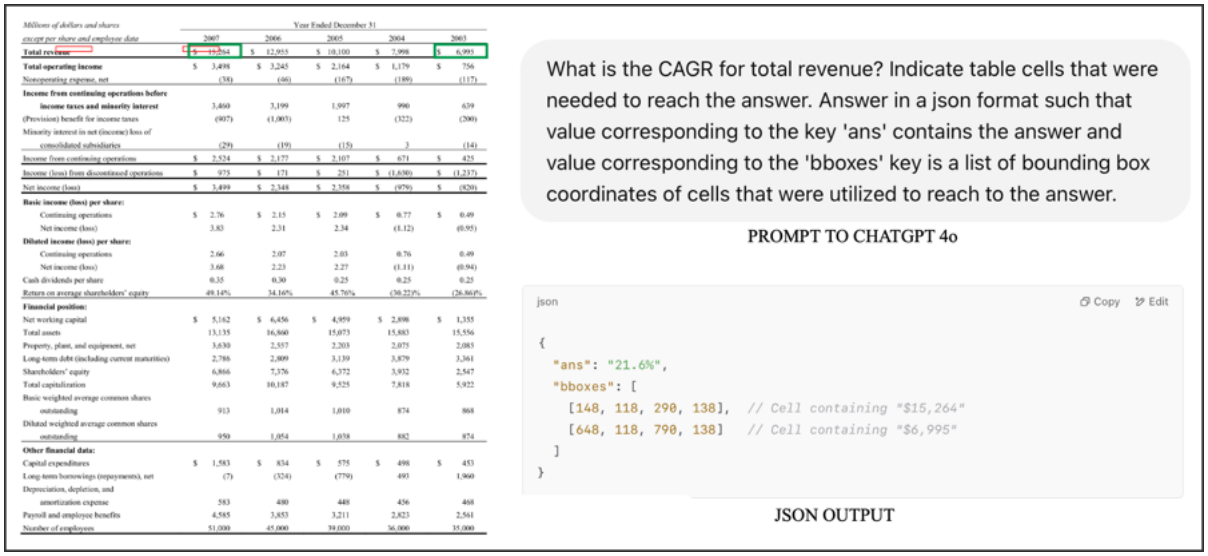


Figure 6.1: Qualitative failure example from a ChatGPT-4^o model. While the answer to the question is computed correctly, the predicted evidence as bounding boxes (red) is misaligned with the true relevant cells (green). Such errors in evidence localization indicate a lack of genuine reasoning over the table structure, highlighting the need for our proposed benchmark.

In summary, the technical motivations for developing EviFiVQA are: (1) to enable rigorous evaluation of VLMs on complex financial documents where both what the answer is and where it comes from matter; (2) to drive research toward models that can perform multi-hop numerical reasoning (as needed for financial calculations) while providing human-interpretable evidence; and (3) to improve the transparency and reliability of AI systems in finance, making them "audit-ready" with traceable outputs. EviFiVQA addresses a crucial gap by providing a large-scale testbed with these characteristics.

Our Contributions: The contributions of this work are as follows:

- **A Novel Financial VQA Dataset with Evidence Annotations:** We present **EviFiVQA**, a large-scale dataset comprising over **1.5 million** question-answer pairs grounded in real-world financial statement tables. Each question is accompanied by its answer and a set of bounding boxes delineating all table cells that serve as evidence for that answer. To our knowledge, this is the first financial VQA dataset of this scale that enforces evidence-grounded answers, enabling evaluation of not just answer accuracy but also the interpretability of model outputs.

- **Diverse Multi-Hop Reasoning Categories:** The questions in EviFiVQA are categorized into five reasoning types of increasing complexity, from simple lookup queries to complex multi-step aggregation and ratio calculations. This taxonomy covers use cases like direct fact retrieval, year-over-year comparisons, multi-year trend analysis, hierarchical summation across subtables, and key-value identification in financial reports. By providing this variety, the dataset tests a model's

ability to handle everything from straightforward extractions to challenging computations that require combining multiple pieces of information across a table or even across subtables.

- **Automated Annotation Pipeline with Expert Validation:** We develop an automated pipeline for generating questions and answers from raw financial tables (leveraging the annotated *FinTabNet* table structures), which ensures scalability to millions of QA pairs. This pipeline programmatically creates questions across the five reasoning categories using template-based generation and then paraphrases them for naturalness. All generated QA annotations are rigorously validated—both through manual sampling by financial domain experts and through automated consistency checks—to ensure a high level of accuracy (with over 97% agreement on correctness during human validation). The result is a reliable dataset suitable for training and testing models in a real-world setting.

- **New Evaluation Protocol for Evidence-Grounded VQA:** We propose a comprehensive evaluation methodology that separately measures answer correctness and evidence localization. In particular, we introduce an *Evidence F1 Score* to quantify how well a model's predicted evidence (set of cells) matches the ground-truth relevant cells, and an *Answer Score* that combines numeric deviation and normalized string similarity (ANLS) for robust answer accuracy assessment. We then define a Combined Score as the product of the two, ensuring that models are only rewarded for answers that are correct *and* justified by correct evidence. This evaluation protocol is tailored to the financial domain's requirement for interpretability and is a contribution that can be applied to other evidence-required VQA tasks.

- **Benchmarking State-of-the-Art Models and Analysis of Results:** We provide extensive experiments evaluating leading large vision-language models (including Qwen2-VL, LLaVA, PixTral, DeepSeek-VL, and others) under both zero-shot and fine-tuned settings on EviFiVQA. Our results reveal that even the best current models struggle with this task, especially in identifying the correct evidence, highlighting a considerable gap between current capabilities and the requirements of financial analytics. We report detailed performance breakdowns by question category, illuminating which types of reasoning are most challenging (e.g., hierarchical aggregation and multi-hop key-value questions) and analyze common failure modes. This benchmark analysis offers insights into shortcomings of existing models, thereby pointing to directions for future research such as improved numerical reasoning, better table structure understanding, and domain-specific pretraining for financial documents.

The rest of this chapter is organized as follows. Before introducing EviFiVQA, we briefly introduce VQAonBD, a competition on VQA on Financial tables that we hosted at ICDAR 2023 conference. We then introduce the EviFiVQA dataset in detail, including the data collection and annotation process, the types of questions and reasoning it encompasses, and the dataset's statistical properties. We also

compare EviFiVQA with prior table QA benchmarks to highlight its unique features. We describe the evaluation protocol we propose for assessing both answer quality and evidence localization. We present our experimental setup and the results of benchmarking several state-of-the-art models on EviFiVQA, along with an in-depth discussion of their performance, error analysis, and qualitative examples. Finally, we conclude the chapter with remarks on the significance of EviFiVQA and potential future directions.

6.2 VQAonBD: A Concise Precursor to EviFiVQA

What it is. *VQAonBD* (`https://ilocr.iiit.ac.in/vqabd/`) is our financial table VQA benchmark built on real business documents. It evaluates whether a system can answer natural-language questions from a table image without requiring explicit evidence localization. We used it to host an ICDAR 2023 challenge and to understand where state-of-the-art pipelines fail *without* grounding—insights that directly motivated EviFiVQA.

Dataset at a Glance. *Source and structure:* Tables come from FinTabNet [4] (10-K/10-Q filings). We recover a cell grid with bounding boxes and normalize common finance-specific quirks (multi-line/stacked headers, horizontal/vertical header splits, and time axes).

Typing and metadata: Each cell is typed (numeric, percentage, currency, parenthetical negative, date/text, etc.), and span information (row/column merges) is retained. This enables principled question generation and arithmetic.

Question families. We generate questions against a *single* table, balanced across five categories:

1. **Lookup** (cell extraction; includes some text answers).
2. **Row-wise ratios** (two columns within a row).
3. **Cross-year ratios/deltas** (temporal comparisons).
4. **Row-wise aggregations across years** (min/max/mean/sum).
5. **Header-driven operations** (semantics from headers such as due within 1 year" vs. after 10 years").

Scale and splits: The corpus is large and mostly numeric: *Train* 39,999 images / 1,254,165 questions (1,197,358 numeric, 56,807 text); *Val* 4,535 / 141,465 (134,651 numeric); *Test* 4,361 / 135,825 (129,861 numeric). Category counts are dominated by lookup and arithmetic over time.

Annotation format: For each table we release (i) a table structure— list of cells with boxes and grid indices; and (ii) a questions answers— list with question id, question text, canonical answer string/number, and answer type. Test-time submissions map question id to predicted answer.

Evaluation Protocol. We compute ANLS (Average Normalized Levenshtein Similarity) and average over questions with max over valid aliases. For numeric answers we also score relative error via a deviation term, then combine with ANLS (L2 norm aggregation). We additionally report exact matches.

ICDAR 2023 Track: Systems in Brief.

- **Upstage KR (ensemble, decompose, extract, compute).** A component-driven pipeline: predict target row/years/operator from the question, answer extractive sub-queries with a strong Category-1 extractor (OCR aided), then apply the operator (ratio/agg). A pseudo-labeling variant improves robustness.
- *NII-TABIQA (structure-first).* Recognize table structure and cell content, normalize headers to a tidy schema (DataFrame), then run a QA module that applies operators over selected cells.
- *DeepSE,×,Upstage-HK (OCR-free generative + arithmetic).* Donut [169]-based model generates candidate values directly from image+question; a small arithmetic head merges/aggregates them.
- *BD-VQA (operator classification + Donut).* Predicts the operator class (ratio/delta/agg), retrieves operands with Donut, computes the final value.
- *SFANC57 (two-stage OCR-free).* Region/value selection followed by aggregation; lighter-weight sequential design.

Leaderboard Summary (Test Set).
Final Combined Score: Upstage KR (**0.959**), NII-TABIQA (**0.901**), Upstage-HK (**0.879**), BD-VQA (**0.640**), and SFANC57 (**0.359**). For reference, a LayoutLM-style baseline [81], [82] performs significantly worse with a score of only 0.163.

Exact Match (Category-Weighted): Upstage KR (**0.931**), Upstage-HK (**0.778**), BD-VQA (**0.397**), NII-TABIQA (**0.374**), SFANC57 (**0.082**), and baseline (**0.015**).

Key Findings (Why This Matters for EviFiVQA).
What works:

1. *Decomposition pays off.* Splitting a complex query into reliable lookups then applying a simple operator yields robust gains across Categories 2–4.

2. *Normalize first, reason second.* Converting images into a clean table schema (header reconciliation, time-axis alignment) reduces ambiguity and boosts arithmetic reliability.

3. *Light arithmetic on OCR-free VLMs is effective.* Donut-like [169] models benefit from a minimal operator head for ratios/aggregations.

Where systems fail:

- *Header semantics and units.* Distinguishing within 1 year" vs. after 10 years" and handling scale notes ("in millions") remain error-prone (notably Cat. 5).

- *Temporal alignment.* Split/stacked year headers lead to operand swaps in cross-year ratios (Cat. 3).

- *String formatting vs. numbers.* Parentheses for negatives, thousand separators, and footnote artifacts depress EM even when numbers are close.

- *Attribution gap.* Without evidence labels, it is hard to diagnose whether errors stem from localization, header parsing, or arithmetic.

Limitations of VQAonBD & Bridge to EviFiVQA.

- *No evidence supervision.* Correct answers may be ungrounded ("right for the wrong reason"); error analysis is opaque.

- *Single-table scope.* Real analyses often span multiple tables/pages; our questions are confined to one table.

- *Units/denominations.* Lack of explicit unit labels allows scale mismatches to sneak through.

Implication: EviFiVQA is designed to *keep* VQAonBD's financial realism and numeric rigor while *adding* explicit evidence grounding: models must return answers *and* supporting cells (boxes), and evaluation couples numeric/string correctness with evidence correctness. This yields transparent, auditable QA and clearer diagnostics for header/temporal parsing and multi-cell reasoning. Our scope complements broader document/table QA benchmarks [36], [170], [171], but focuses on SEC-style financial statements where numeric reasoning dominates. VQAonBD established that (i) simple decomposition + reliable extraction + lightweight arithmetic achieve high accuracy (0.96 combined; 0.93 EM) on financial tables; and (ii) the absence of grounding is now the main bottleneck. EviFiVQA addresses this by making evidence *first-class* in both data and metrics.

6.3 EviFiVQA Dataset

We construct the EviFiVQA dataset from publicly available S&P 500 company annual reports (specifically, the financial statements in 10-K filings) spanning years 1999–2019, leveraging the table annotations provided by the FinTabNet benchmark [4]. FinTabNet consists of a large collection of financial tables extracted from PDF documents, with detailed structural markup (identifying rows, columns, and cell coordinates). These multi-page PDF reports intermix textual discussion with tabular data, and FinTabNet's table structure annotations allow us to automate the creation of a Visual Question Answering dataset that is explicitly evidence-based. In other words, using FinTabNet as a foundation, we generate questions that refer to the content of tables, and we use the known table structures to identify which cells contain the information needed to answer each question. Each question-answer pair in EviFiVQA is thus annotated with the exact bounding boxes of all relevant table cells, ensuring that any model evaluated on this dataset must provide answers that are *traceable and interpretable* via those evidence boxes.

The financial tables in FinTabNet (and hence in EviFiVQA) pose significant visual and structural challenges due to their high row/column density, irregular layouts, and hierarchical grouping of entries. Often, a single financial statement will contain multiple subtables or sections, sometimes with different column spans, or separate segments for different currencies or units. Figure 6.2 shows examples of such complex layouts from our dataset, including instances where tables are horizontally or vertically split into subtables and cases where multiple units (e.g., thousands vs. millions) appear in different parts of the same statement. Additionally, the core financial statements (balance sheets, income statements, cash flow statements) exhibit a tree-like hierarchical organization: broad categories (e.g., *Assets*, *Liabilities*, *Revenue*, *Expenses*) form the top-level groups, which are broken down into subcategories and further into line-item entries. This hierarchy is often indicated visually by indentation or grouping and by subtotals that aggregate lower-level entries. Such structure facilitates analysis like computing totals or identifying the contributions of subcomponents. Consequently, EviFiVQA covers a spectrum of question complexities, from simple fact-finding to queries that require understanding the table's hierarchical relationships and performing multi-step operations (e.g., summing across a subset of rows that constitute a category total).

6.3.1 Dataset Annotation Process

Our dataset construction employs a semi-automated annotation framework that extracts the content and structure of each table, and then generates diverse question-answer pairs grounded in that structure. The goal is to ensure that for every question we create, we can unambiguously determine the correct answer from the table and also know exactly which cells in the table are involved. Figure 6.2 already illustrated some raw table examples; we now describe how we generate QA pairs from such tables.

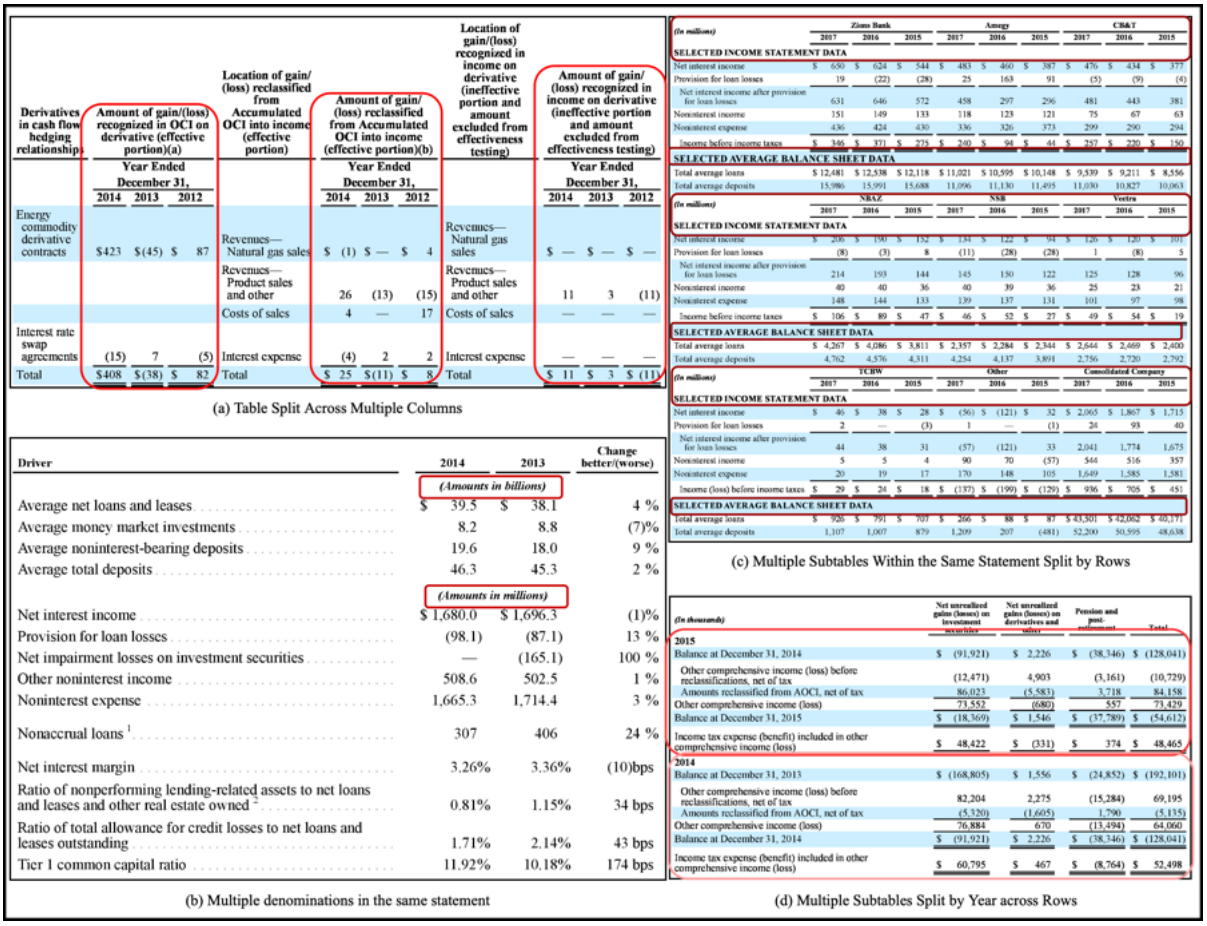


Figure 6.2: Examples of complex table layouts from the EviFiVQA dataset. These include (left) horizontally split subtables, (middle) vertically concatenated tables, and (right) statements containing values in multiple units (denominations) on the same page. Such irregular and rich layouts, inherited from real financial reports, pose challenges for automated reasoning and evidence localization.

At a high level, the process has four main stages: (1) **Cell Type Classification**, (2) **Structural Analysis of the Table**, (3) **Hierarchical Tree Extraction**, and (4) **Question Generation**. Pseudocode for the annotation pipeline is provided in Algorithm 6. We will walk through these steps with additional detail:

In Step 1 (*Cell Type Classification*), we examine the text content of each cell (obtained via OCR or via the FinTabNet annotation if provided) to categorize it. This helps later steps; for instance, knowing which cells contain years (e.g., "2008") allows us to identify column headers for multi-year tables, and knowing which cells are purely textual vs. numeric helps distinguish row labels from data cells. We use simple heuristics and pattern matching for this classification (e.g., a cell matching `\d{4}` is likely a year, a cell with a percent sign is percentage, etc.).

Algorithm 6 Dataset Annotation Pipeline

Input: Financial table image (with FinTabNet structure annotations)
Output: Annotated set of QA pairs with answers and evidence bounding boxes
Step 1: Cell Type Classification
Classify each table cell's content type as one of: year, date, numeric, percentage, empty, text.
Step 2: Structural Segmentation
Identify major table components:
– Locate column header rows and isolate their cells.
– Determine where the main data region of the table begins (e.g., first row that is not a header).
– Use content patterns (year formats, presence of currency symbols, etc.) to differentiate columns.
– Classify each row as either a header row, a subtotal/total, or a regular data row.
(Sub-step A: Identify Column Headers)
If multiple header rows exist, merge them appropriately. Mark columns with multi-year data versus single-year data.
(Sub-step B: Identify Table Hierarchy)
Detect hierarchical grouping of rows by scanning for patterns:
– Find instances where a numeric entry in one row is the sum of a set of subsequent rows (indicating a subtotal).
– Use textual cues (indentation levels of row labels, or specific keywords like "Total" or "Net") to infer parent-child relationships among rows.
– Build a tree representation where each node is a financial line item and children are its components.
(Sub-step C: Identify Row Headers)
Using the hierarchy and indentation:
– Group consecutive rows with similar indentation as part of the same section.
– Associate group headers (e.g., a bold line item that appears to be a category name) with the following indented entries.
– Mark final totals that correspond to grouped sections.
Step 3: Integrity Checks and Pruning
Normalize any detected groupings and remove spurious hierarchy links. Prune any groupings below a confidence threshold (to avoid including incorrect relationships).
Step 4: Question & Answer Generation
For each reasoning category (1 through 5, defined in Section 6.3.2), generate one or more question templates and fill in placeholders with actual table content:
– Create direct lookup questions (Category 1) by selecting individual cells (ensuring unique identification by row and column labels).
– Create ratio questions (Category 2) by picking two related figures (e.g., same row, two years) and forming a "what is the ratio of X to Y" query.
– Create all-years aggregation questions (Category 3) by using an entire row of numeric values across years (compute sum, average, min, max, or CAGR).
– Create hierarchical aggregation questions (Category 4) by selecting a parent node in the hierarchy and asking for an aggregate of its children (sum or average of sub-items).
– Create key-value extraction questions (Category 5) by identifying the largest (or smallest) contributor among a set of siblings in a hierarchy, or asking for the value of a specifically named sub-item.
Use a pretrained language model (e.g., LLaMA 3.1 7B) to paraphrase the templated questions for variety and fluency, while ensuring the meaning is preserved.
Return: Structured dataset of QA pairs, each with: the question (natural language), the answer (numerical or textual, as appropriate), and the list of bounding box coordinates of all cells used to determine that answer.

In Step 2 (*Structural Segmentation*), we leverage the structural hints from FinTabNet (which gives us cell bounding boxes and an initial table structure tree) and refine them. The table's first few rows often contain column headers (like years or currency units), which we detect by their content (years or keywords like "USD"). We then determine the main data region—where actual financial line items begin. By parsing cell contents, we also detect patterns such as sequences of years across columns, which signals a multi-year table, or presence of text in what should be numeric columns, which might indicate subheaders or footnotes.

A crucial part of this step is identifying the inherent **hierarchical structure** of the table (Sub-steps B and C). Financial statements often list line items in a hierarchy. For example, an income statement might have a section with a parent line "Net Income" and several indented child lines like "Operating

Income," "Other Income," etc., whose sum equals the parent. We algorithmically detect such relationships by looking for additivity: if a number in a row equals the sum of a set of numbers in subsequent contiguous rows, we infer that the row is a subtotal of those. Words like "Total" or "Net" in a row label strongly indicate a subtotal/total row. Indentation levels (which we estimate from cell coordinates, since an indented label is usually slightly to the right of a higher-level label above it) further indicate grouping: a more indented row is likely a child of a less indented one above. By combining these clues, we build a tree where each node has children that sum up (in whole or in part) to it. We take care to remove redundant or nested groupings that conflict (for instance, if a small subset of rows is identified as summing to an intermediate total, and those plus other rows sum to a higher total, we keep the larger grouping and discard any duplicative smaller structure that is encompassed by it).

After structural analysis, Step 3 performs integrity checks to ensure the inferred structure makes sense globally. We verify, for example, that every opening "Total X" has corresponding components summing to X's value, within a tolerance. If any inconsistencies are found (like due to minor OCR errors or unusual table formatting), those are flagged and that part of the table is skipped for question generation to avoid erroneous QA pairs.

Finally, Step 4 (*QA Generation*) uses the structured information to generate questions in each of the five categories of reasoning that we target. We have designed question templates for each category. For example, for Category 1 (Extractive), a template might be: *"According to the report, what is the value of [Row Label] in [Year]?*" which we would instantiate with a specific row (say, "Net Revenue") and a specific year (e.g., 2008) that exist in the table. We ensure that the combination is unique (there should be a single cell at the intersection of that row and column to answer the question). For a Category 2 (Ratio-Based) question, a template could be: *"What is the ratio of [Row Label] in [Year1] to [Year2]?"*, for which we pick two years and a row such that both year columns have a numeric value for that row. We then compute the ratio as the answer. We take similar approaches for other categories: for Category 3 (All-years Aggregation), we might generate a question asking for the average or maximum of a particular row's values across all years present; for Category 4 (Hierarchical Aggregation), we identify a subtotal line (like "Total Assets") and ask something like *"What is the sum of all components contributing to [Total Assets] for [Year]?"* which essentially checks if the model can identify and add the child values; for Category 5 (Key-Value Extraction), we could ask *"Which item had the highest value in [Category] in [Year], and what was that value?"*, requiring identification of the largest sub-item under a category for a given year.

To avoid the questions sounding too templated or monotonous, we use a large language model to paraphrase each generated question. For example, the template question *"What is the net revenue of the organization in 2008?"* might be paraphrased to *"In 2008, what did the company report as its net revenue?"* The paraphrasing model (we used a LLaMA 3.1 7B variant fine-tuned for instruction following)

was guided with prompts to preserve the meaning and specificity while varying the phrasing. We found that paraphrasing not only makes the dataset linguistically richer (more like natural human questions) but also occasionally surfaces ambiguities, which we then could correct (for instance, if a paraphrase made a question unclear, we adjusted prompts or dropped that question).

Throughout this automated generation, we maintain a mapping from each question to the set of table cells used in computing the answer. This mapping is straightforward for direct lookup questions (just one cell) and for hierarchical or multi-step questions (multiple cells). The bounding box coordinates of each such cell (relative to the page/image) are recorded using the FinTabNet coordinates.

After these steps, we obtain a massive collection of QA pairs. The next subsections detail the nature of the questions generated (categorized by reasoning type), and the validation procedure ensuring their quality.

6.3.2 Question Types and Reasoning Categories

One of the distinctive features of EviFiVQA is that it categorizes queries by the type of reasoning required. We defined five broad categories of questions, arranged in roughly increasing order of complexity. This categorization not only helps in analyzing model performance (to see which kinds of reasoning are hardest) but also in guiding the question generation process. The categories are defined as follows (with examples in italics):

Category 1: Extractive (Direct Lookup): These are straightforward questions where the answer is a single value that can be read directly from one cell of the table, after identifying the correct row and column. For example, *"Based on this financial report, what is the net revenue of the organization in 2008?"* requires finding the cell at the intersection of the "Net Revenue" row and the "2008" column. No calculation is needed beyond possibly reading the number (the answer is explicitly present in the table). This category tests a model's ability to locate information by understanding row/column labels.

Category 2: Ratio-Based Reasoning: These questions require computing a ratio between two numerical values in the table, often from the same row across two columns or between two related rows. For example, *"What is the ratio of the net revenue in 2007 to that in 2008 for this organization?"* expects the model to find the net revenue for 2007 and 2008 and then divide the former by the latter. This involves two lookup operations and a division. Such questions mimic financial analysis tasks like year-over-year growth (which is essentially a ratio minus 1).

Category 3: All-years Aggregation: These questions demand an operation (sum, average, minimum, maximum, or a compound growth rate) applied over multiple years for a single financial metric (row). For example, *"For this organization, what is the compounded annual growth rate (CAGR) of net income across all the years presented?"* or *"What was the average total assets over the period 2015–2019?"* To answer, a model must gather all values of the specified row across the specified range

of years and perform the requested calculation. This category introduces multi-step arithmetic and an understanding of sequences.

Category 4: Hierarchical Aggregation: These questions involve a table's hierarchical structure. The model must aggregate values of a set of child entries to yield a parent value, or vice versa. For instance, *"What is the total expenditure for 2008, considering all its contributing sub-categories?"* might refer to a case where "Total Expenditure" is the sum of several line items (like R&D, Marketing, etc.). The model must recognize which line items sum up to "Total Expenditure" (using the hierarchy) and either verify a given total or compute it if not explicitly in the table (though usually totals are present). Another example is *"What is the average value of all the factors contributing to net income for 2008?"*, where it must identify all components that roll up into net income for that year and average them. This category tests multi-hop reasoning across rows that are not adjacent but linked by financial semantics.

Category 5: Key-Value Extraction: This is a more complex multi-hop reasoning category where the question asks for a qualitative insight along with a quantitative value. Typically, these questions ask for an identifying key (a particular sub-item) and its value. For example: *"Which factor contributed the most to net income in 2008, and what was its value?"* Here the model needs to look at all components under "Net Income" for 2008, find which one has the largest value, and then report the name of that component (the key) and the value. This requires understanding of both hierarchy (to know what the "factors contributing to net income" are) and comparison (to find the maximum). It is effectively a two-part question.

Each question in the dataset is linked to the ground-truth bounding boxes of the cells required to answer it. If a question is Category 1 (extractive), this is usually just one cell's box. For Category 2 (ratio), it would be two cells (the numerator and denominator values). For Category 3 (all-years aggregation), it could be multiple cells across a row. Category 4 (hierarchical) would involve all the child cells that need to be summed or averaged (and possibly the parent cell as well if present). Category 5 (key-value) would involve the set of sibling cells being compared and the one that is the answer. By providing these annotations, we enable precise evaluation of a model's evidence selection ability.

Figure 6.3 provides a visual illustration of example questions from each category for a sample financial statement image. This figure demonstrates how different questions "pick apart" the information in various ways: e.g., a Category 1 question might highlight a single cell (say, 2010 revenue), whereas a Category 4 question could highlight an entire group of cells corresponding to components of a total. Such visualizations emphasize that higher-category questions often require gathering evidence from multiple disjoint parts of the table.

We refer the reader to Table 6.1 for the distribution of questions across these categories in our dataset, and to Figures 6.4 and 6.5 for further statistical insights (such as how many cells are typically involved in

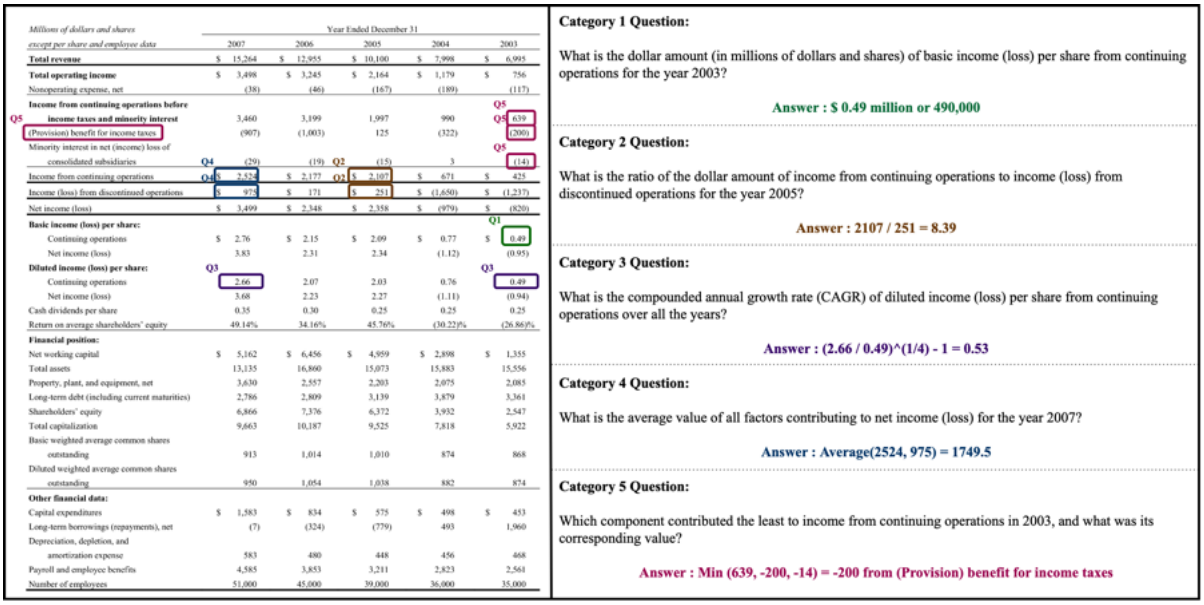


Figure 6.3: Examples of different question types from a single financial statement in EviFiVQA. The image shows an excerpt of a table with highlighted cells for each example question. **Category 1:** Direct lookup (green) asks for a value from one cell. **Category 2:** Ratio-based (blue) requires dividing values from two cells. **Category 3:** All-years aggregation (purple) involves multiple cells across a row. **Category 4:** Hierarchical aggregation (orange) highlights child entries that sum to a parent. **Category 5:** Key-value (red) requires finding the largest contributor among siblings.

answering questions, and the distribution of question lengths). Broadly, about one-third of our questions are simple lookups (Category 1), while the remaining two-thirds require some form of computation or reasoning beyond a single cell. This balance ensures that while models must master basics of table navigation, they are also heavily tested on advanced reasoning.

6.3.3 Dataset Validation and Quality Assurance

Building a dataset of this scale (1.5M QA pairs) required automation, but it was vital to ensure accuracy to avoid training or evaluating models on faulty data. We adopted a two-pronged validation strategy: *human verification* on a sample of the data, and *automated consistency checks* across the entire dataset.

To facilitate human validation, we randomly selected 2 to 5 questions per document image. We focused specifically on categories 3 and 5 due to the higher number of row (Category 5) and column (Category 3) cells within certain sections of those documents. Given that our dataset spans 42,387 images (financial statement pages), this procedure yielded on the order of 42,387 $\times$ (somewhere between 2 and 5) $\approx$ 150,000 QA pairs checked by humans (which is a manageable though not trivial number). Each sampled QA pair was examined by a subject matter expert (SME) with experience in finance or

accounting. The SME would cross-verify the annotated answer and the highlighted evidence bounding boxes against the original document image to ensure two things: (1) the question was answered correctly (i.e., the answer value is indeed the correct solution to the question given the table data), and (2) the evidence annotation was correct and complete (i.e., all needed cells were highlighted, and no extraneous cells were incorrectly included). If either the answer or evidence was found to be wrong or ambiguous, that was marked as an error case.

This manual review found a **97.37% agreement** between the SME evaluations and our automatically generated annotations, indicating a high level of accuracy in the dataset. In other words, over 97% of the sampled QA pairs were deemed entirely correct. For the remaining 2.6%, we closely inspected the errors to categorize their causes:

- About 1.2% of all cases (roughly half of the errors) were due to ambiguity introduced by the paraphrasing of the question. For instance, a paraphrased question might use phrasing that could be interpreted in more than one way (e.g., referring to "growth rate" without clarifying the period). In such instances, either the question could match multiple possible answers or was confusing to the validator. We resolved this by either re-generating or clarifying those questions in the dataset (or dropping them if irredeemably ambiguous).
- Approximately 0.4% of cases had ambiguity or errors in the answers because of multi-level numerical hierarchies. For example, if a question asks for a sum of components, and there were sub-sub-components, our automation might have double-counted or been unsure which level to aggregate. We refined the hierarchy extraction code to handle these better, or in a few edge cases, removed problematic questions.
- The remaining 1% of cases were boundary issues not properly handled by automation, such as a question inadvertently including a footnote cell as part of evidence or failing to exclude a zero-value that should have been ignored. We documented and fixed these specific logic issues in our pipeline, then regenerated those parts of the dataset.

In addition to the manual checks, we deployed automated validations at scale:

- We cross-verified that every annotated evidence cell in a QA pair actually contains the value used in computing the answer. Since FinTabNet provides ground-truth content for cells, we could, for numeric answers, recompute the expected result from the annotated cells and compare it to the stored answer. For textual answers, we could check that the answer string matches the content of at least one annotated cell (or a combination, in cases where an answer is concatenated text from multiple cells).
- We ensured that the bounding box coordinates we provide for evidence align with the original FinTabNet annotations for those cells (to be sure we didn't accidentally shift a box or assign it

	Num Images	Category 1	Category 2	Category 3	Category 4	Category 5	Total
Training Set	34,790	481,025	285,881	200,482	117,138	165,471	1,249,997
Test Set	3,654	49,386	30,127	21,936	11,678	15,137	128,264
Validation Set	3,943	53,352	31,590	22,583	12,566	16,983	137,074
Total	42,387	583,763	347,598	245,001	141,382	197,591	1,515,335

Table 6.1: Distribution of questions across varying difficulty categories.

to the wrong cell). All evidence boxes correspond to actual table cells delineated in the original PDFs.

- For arithmetic questions, we recalculated sums, differences, ratios, and growth rates from raw cell values and confirmed they equaled our annotated answers to within a very small tolerance (we allowed minor rounding differences for things like ratios and CAGR when expressed in percentage form).
- We also applied a semantic consistency check for paraphrased questions. Using a sentence embedding model (Sentence-BERT [172]), we measured the similarity between each original template question and its paraphrased version. We flagged any paraphrased question with a similarity score below 0.85 (on a 0–1 scale) for manual review, to ensure that the meaning hadn't drifted. In practice, the vast majority of paraphrases were fine, and the flagged ones were typically edge cases where wording changed some nuance (those were corrected or the question dropped).

Any QA pair that failed an automated check was either automatically corrected (if possible) or removed from the dataset to be safe. This dual approach of human and automated validation helped us achieve a high confidence in the dataset's quality. The negligible error rate means that model performance on EviFiVQA will reflect the model's capabilities rather than artifacts of incorrect ground truth.

6.3.4 Dataset Statistics and Characteristics

Beyond quality, it's important to understand the composition of EviFiVQA. Table 6.1 provides a summary of the dataset's size and breakdown. We have a total of 1,515,335 question-answer pairs, split into 1,249,997 for training, 137,074 for validation, and 128,264 for testing. The split is by documents: no financial statement appears in more than one set, ensuring that evaluation is on held-out companies or years.

Each document (financial statement page) on average contains about 30–40 questions in our dataset (some have more, especially if the page has multiple tables or a very dense table). The distribution of questions across the five categories in each split is also given in Table 6.1. Roughly speaking, Category 1 (extractive) constitutes the largest portion, followed by Category 2 and 3, etc., with Category 5

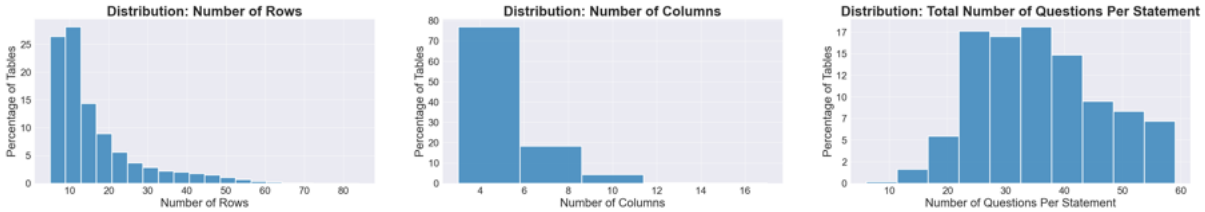


Figure 6.4: Distributions of key parameters at the document (image) level in EviFiVQA. **Left:** Histogram of the number of rows in tables (many statements have on the order of tens of rows, with some very large tables reaching over 100 rows). **Middle:** Distribution of number of columns per table (most commonly 3–6, corresponding to multi-year financial statements with perhaps a total column). **Right:** Distribution of the number of QA pairs generated per document. Most documents yield a few dozen questions, with some very information-dense pages resulting in 50 or more questions.

(key-value) being the smallest portion (but still tens of thousands of questions). This was by design: extraction is common and we need a strong base of those questions, but we also wanted a significant representation of harder reasoning to ensure models don't overfit to simple lookups.

To further characterize the dataset, Figure 6.4 depicts some image-level statistics: the number of rows per table, number of columns per table, and number of questions per statement (image). We observe that while many tables have a moderate number of rows (20–40) and columns (4–8, typically years plus maybe a total column or similar), there is a long tail of very large tables (some financial statements list over 100 line items, especially if they combine multiple periods or segments). The question count per image naturally correlates with table size.

Figure 6.5 presents question-level distributions: (i) how the questions are divided among the five categories (by percentage), (ii) the number of distinct table cells typically required to answer a question (for evidence, i.e., evidence set size), and (iii) the distribution of question lengths in tokens. The evidence set size is particularly interesting: for Category 1 questions, this value is 1 by definition. For ratio questions, it's usually 2. For Category 3, it can be several (however many years are present; our dataset's tables commonly have 5–10 years of data, so often around that many cells). Category 4 might involve anywhere from 2 or 3 cells (if summing two children) up to 10 or more (if a top-level total has many components). Category 5 typically involves checking all siblings of a group—often 3–6 items—to find the max/min, so evidence size includes those siblings plus possibly the one identified. The distribution of evidence cell counts thus shows a peak at 1 (mostly Category 1 questions), another around 2 (Category 2 and some Category 4 with just two items in a sum), and a tail extending beyond 5 for more complex queries. Meanwhile, question length tends to fall in a moderate range (most questions are 10–15 words after paraphrasing), which is beneficial for current language models to parse easily.

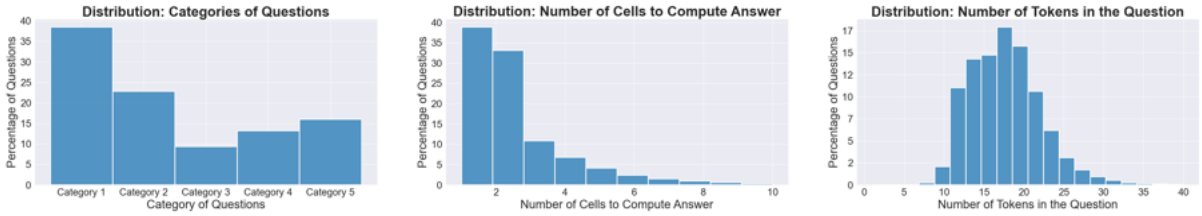


Figure 6.5: Distributions of question-level parameters in EviFiVQA. **Left:** Proportion of questions belonging to each reasoning category (1 through 5). **Middle:** Distribution of the number of table cells that a question's answer depends on (i.e., evidence cells). While many questions require 1–2 cells (direct lookups or simple ratios), a significant fraction require 3 or more cells (aggregations and hierarchical reasoning). **Right:** Distribution of question lengths (in tokens/words) showing that most questions fall in the range of 5 to 20 words, reflecting natural language variability introduced by paraphrasing.

6.3.5 Novelty of EviFiVQA

Existing datasets for question answering on tabular data (including those in the financial domain) have provided valuable stepping stones, but EviFiVQA introduces several unique features that differentiate it from prior work. Table 6.2 offers a comparison among prominent Table QA benchmarks. Notably, FinQA [35] and the very recent TableVQA-Bench (sometimes referred to as FinTabNetQA in literature) [173] are closest to our domain. FinQA provides about 8.5K expert-annotated QA pairs over financial documents, focusing on numerical reasoning with both textual and tabular evidence. However, FinQA primarily emphasizes reasoning over extracted data and textual reports and does not require models to output the *location* of evidence in the document image. Moreover, FinQA's scale is much smaller, and it doesn't cover the visual layout challenges since it assumes structured tables as input. TableVQA-Bench, on the other hand, builds on FinTabNet's tables by generating questions via large language models (similar spirit to our approach) and rendering tables as images for a vision-model evaluation. While TableVQA-Bench addresses visual understanding and includes multiple table domains, it lacks manual validation and crucially, it does not explicitly annotate the underlying *tree structure* of tables or require evidence localization. It evaluates accuracy of answers but not whether the model can highlight where the answer came from.

In contrast, EviFiVQA uniquely integrates the inherent hierarchical tree structures of financial tables directly into the question formulation (especially for categories 4 and 5). By doing so, our dataset enables and requires models to perform complex multi-hop reasoning that mirrors how an analyst might traverse a financial statement: understanding that "Net Income" is composed of "Operating Income" and "Non-operating items" etc., or that "Total Liabilities" is the sum of current and long-term liabilities, and so on. This is a capability not explicitly exercised by previous datasets. Additionally, each EviFiVQA question is coupled with the precise bounding boxes of all relevant cells. This explicit evidence annotation means that any model trained or evaluated on our dataset must develop a form of grounded reasoning—essentially aligning its latent reasoning steps with human-interpretable evidence on the doc-

Table 6.2: Comparison of Table QA Datasets

Dataset (Year)	Size	Source	Annotation	Unique Points
WikiTable-Questions (2015)	2K tables; 22K QA	Wikipedia (HTML)	Crowdsourced (Natural Qs)	Compositional, multi-step
WikiSQL (2017)	120K (tables + SQL)	Wikipedia	Crowdsourced (NL to SQL)	SQL generation, conditions + aggregations
TabFact (2020)	16.5K	Wikipedia	Auto-gen + human check	Fact verification (True/False), logical reasoning
HybridQA (2020)	70K QA	Tables and text	Crowdsourced (multi-hop)	Combines table + multi-hop reasoning
TAT-QA (2020)	16K QA	10-K financials	Experts and Crowdsourced	Numeric multi-step, real-world finance
FeTaQA (2021)	10.3K QA	Wikipedia (semi-structured)	Crowdsourced (free-form)	Longer answers, multi-entity aggregation
FinQA (2021)	8K QA	Earnings reports and FinTabNet	Finance experts	Multi-step numerical calculations
AIT-QA (2022)	10-K forms (5 airlines)	SEC EDGAR	Human (airline KPIs)	Airline-specific tables, KPI queries
TableBench (2023)	886 + 19.7K QA	Real-world tables	ChatGPT and Human checked	Numeric + multi-domain coverage
***EviFiVQA** (2025)*	***1.5M QA (5 categories) + Evidence***	***FinTabNet***	***Automated with Expert Validation***	***Semantic, multi-step complex aggregations***

ument. This level of interpretability is especially critical for financial applications (as discussed earlier) but was not present in prior Table QA benchmarks.

Table 6.2 highlights how EviFiVQA compares to related datasets on key aspects such as size, source, annotation method, and unique features. EviFiVQA is an order of magnitude larger than most, uses real financial filing data (like FinQA) but in a fully visual context, and provides a combination of large-scale automated generation with expert validation (bringing together scalability and quality). The questions in our dataset often require reasoning across *both* axes of the table (rows and columns) simultaneously and dealing with irregular layouts—something like WikiSQL (with its reliance on structured tables and SQL queries) or WikiTableQuestions (which deals with semi-structured wiki tables but not complex financial ones) did not cover. In short, EviFiVQA is a novel benchmark that fills an important gap for evaluating vision-language models in a domain where the answers must be correct, explainable, and directly linked to original content.

6.3.6 Comparison with Prior Benchmarks

To put the scope and difficulty of EviFiVQA into perspective, we compare it qualitatively with the most relevant prior datasets from Table 6.2. FinQA [35] and FinTabNetQA/TableVQA-Bench [173] are particularly noteworthy. FinQA's questions require reasoning over financial data but rely on both textual and tabular data and do not enforce evidence localization or visual understanding (tables are given in structured form). FinTabNetQA, derived from FinTabNet, rendered tables as images and generated questions automatically (somewhat similar to our approach), but it does not incorporate the hierarchical tree structure of tables into the question logic. It also lacks manual validation of each QA pair and does not provide ground-truth evidence for answers. In EviFiVQA, every question is grounded in the table's structure — for example, when we ask a hierarchical question, we ensure that the model must identify the exact sub-components (with their bounding boxes) that produce the answer. This explicit grounding in evidence is a new challenge for models.

Furthermore, EviFiVQA's scale (1.5M QA) far exceeds FinQA (8K) and FinTabNetQA (which, while it has 20K QA, those are divided among a variety of domains; only a fraction are financial). This scale allows training large models without overfitting and facilitates probing for emergent behaviors or fine-grained category performance. We believe that the introduction of EviFiVQA will drive progress in developing VQA systems that are not just accurate but also interpretable — a necessary step for adoption in real financial analytics pipelines where decisions must be audited and justified.

6.4 Evaluation Protocol

A core contribution of our work is a tailored evaluation protocol for evidence-grounded VQA on financial tables. Traditional VQA accuracy metrics (exact match, F1, etc.) focus solely on whether the answer is correct. In our setting, we have a dual objective: the model must produce the correct answer *and* identify the correct supporting cells. We therefore split the evaluation into two components: an **Answer Score** and an **Evidence Score**, and then combine them into a single overall metric that reflects performance on both dimensions.

In particular, the *Evidence Score* measures how accurately the system detects all the necessary table cells (by their bounding boxes) that support the answer, and the *Answer Score* assesses the final predicted answer's correctness in a way that is forgiving of minor OCR errors or formatting differences (since the answer is derived from possibly imperfect text extraction). We then define a combined score that effectively ensures a high score is only achieved if both answer and evidence are correct.

6.4.1 Evidence Score

To evaluate evidence localization, we consider the set of ground-truth relevant cells for a question and the set of cells predicted by the model. Let $\mathcal{G} = g_1, g_2, \ldots, g_{|\mathcal{G}|}$ be the set of ground-truth bounding boxes (each g_i representing one table cell's bounding box on the page) that contain the information needed for the answer. Let $\mathcal{P} = p_1, p_2, \ldots, p_{|\mathcal{P}|}$ be the set of bounding boxes predicted by the model as its evidence output. The task is essentially to measure the overlap between $\mathcal{P}$ and $\mathcal{G}$, but because these are spatial coordinates, we need to define what it means for a predicted box to match a ground-truth box.

We use the standard **Intersection-over-Union (IoU)** metric to determine if a predicted evidence box corresponds to a ground-truth evidence box. For any predicted box p and ground-truth box g, IoU is defined as:

$$\mathrm{IoU}(p, g) \;=\; \frac{|p \cap g|}{|p \cup g|},$$

where $|p \cap g|$ denotes the area of overlap between the two boxes and $|p \cup g|$ denotes the area of their union. This yields a value between 0 and 1, with 1 indicating perfect overlap and 0 indicating no overlap.

Using IoU, we then perform a one-to-one matching between the predicted and ground truth sets:

1. For each predicted box $p \in \mathcal{P}$, find the ground-truth box $g^* \in \mathcal{G}$ that has the maximum IoU with p. Let $\mathrm{IoUmax}(p) = \max g \in \mathcal{G}\mathrm{IoU}(p, g)$.

2. We set a threshold δ (we use $\delta = 0.5$ in our evaluations) such that if $\mathrm{IoU}_{\max}(p) \geq \delta$, we consider p a correct detection of the ground-truth box g it overlaps with. We mark g as matched and p as a True Positive (TP).

3. If a predicted box p does not reach IoU δ with any unmatched ground truth box, then p is considered a False Positive (FP) (it either overlaps only with some already matched ground truth, or it doesn't sufficiently overlap with any ground truth).

4. After processing all $p \in \mathcal{P}$, any ground-truth box in $\mathcal{G}$ that remains unmatched is counted as a False Negative (FN).

This matching mechanism is analogous to evaluating object detection or segmentation: a predicted evidence box is "correct" if it significantly overlaps with a ground-truth evidence region (and we ensure each ground-truth can only be matched once). The threshold $\delta = 0.5$ is a moderate criterion that allows for slight misalignments (which are expected, as a model might not output coordinates that exactly match the annotation) but requires the overlap to be more than trivial.

Once we have TP, FP, and FN counts for evidence selection on a question, we define **Evidence Precision (EP) & Evidence Recall (ER):**

$$\mathrm{EP} \;=\; \frac{\mathrm{TP}}{|\mathcal{P}|}, \quad \mathrm{ER} \;=\; \frac{\mathrm{TP}}{|\mathcal{G}|}.$$

Precision (EP) measures how much of the model's predicted evidence is actually relevant (penalizing inclusion of extraneous cells), while recall (ER) measures how much of the true relevant evidence the model managed to find (penalizing missing any required cell).

We then combine these into an **Evidence F1** score:

$$\mathrm{EvidenceF1} \;=\; 2 \;\times\; \frac{\mathrm{EP} \times \mathrm{ER}}{\mathrm{EP} + \mathrm{ER}}.$$

which is the harmonic mean of evidence precision and recall. The EvidenceF1 is 1.0 only if the model found all and only the relevant cells (perfect precision and recall). It is 0 if the model's evidence predictions have no overlap with the ground truth at all. By using F1, we balance the model's ability to cover all needed evidence with its tendency to possibly include extra cells. In financial QA, including extraneous cells as evidence is problematic because it could indicate the model is bringing in irrelevant information; missing a needed cell is also problematic because it means an answer is not fully justified. Thus, EvidenceF1 is a suitable single measure of evidence localization performance.

We note that we intentionally chose a moderately forgiving IoU threshold (50%). We experimented with higher thresholds like 0.8 or requiring exact matches, but due to minor differences in how a model might draw box boundaries, a too-strict criterion could unfairly penalize a nearly correct evidence selection (e.g., a box that covers 90% of the cell but misses a margin). At 0.5 IoU, we give credit as long as the predicted region significantly overlaps the correct cell. The model still has incentive to refine its output because if it spuriously merges two cells into one big box, precision would drop (since one predicted box would only match one ground truth, leaving the other unmatched, thus a FP and a FN).

Some questions in EviFiVQA require multiple evidence cells (for example, a ratio question needs two cells). For such cases, partial credit is given: if the model finds one of the two correctly, it will have an EvidenceF1 less than 1 but greater than 0. We aggregate EvidenceF1 across all questions in an evaluation set by simply taking the average (since each question's evidence retrieval is an independent event, and we want an overall measure).

6.4.2 Answer Score

For the answer itself, we cannot simply use exact string match or exact numerical match, because models might express the answer in a slightly different way (especially if there's OCR noise or format differences). We adopt an approach that considers both numeric deviation and token similarity, building on metrics used in recent document VQA challenges (notably, the TextVQA and DocVQA communities

often use *Normalized Levenshtein Similarity*, ANLS).

During evaluation, the model receives only the document image (with no additional structured input) and the question text, and it must produce an answer. We denote the model's predicted answer as $\widehat{o}q$ for question q, and the ground truth answer(s) as ai (some questions might have multiple acceptable answers, e.g., a monetary value might be acceptable with or without a currency symbol). We compute scores at the level of each question and then average over all questions to get dataset-level performance.

We consider two sub-components for answer evaluation:

1. **Numerical Deviation Score**: If the answer is or contains numeric values, we want to give partial credit if the predicted number is close to the correct number (for instance, predicting 99 when the answer is 100 should be better than predicting 50). We define this formally for pure numeric answers.

2. **Averaged Normalized Levenshtein Similarity (ANLS)**: This measures the textual similarity between the predicted answer string and the ground truth answer(s), in a way that handles minor OCR mistakes or wording differences.

Averaged Absolute Deviation (for numeric answers): Let x be the correct numeric value and $\widehat{x}$ be the model's predicted numeric value for a given question. We define a deviation-based score:

$$\text{DeviationScore} = \begin{cases} 1 - \frac{|\widehat{x}-x|}{|x|}, & \text{if } |\widehat{x}-x| < |x|,\ 0, \\ \text{if } |\widehat{x}-x| \geq |x|, \end{cases} \tag{6.1}$$

i.e., we linearly map the absolute error $|\widehat{x} - x|$ to a [0,1] score by comparing it to the magnitude of x. If the error is 100% or more of the true value, we give 0 credit (because being off by that much is essentially a completely wrong answer). If the prediction is exactly correct, $|\widehat{x} - x| = 0$ and the score is 1. If the prediction is, say, 10% off, the score would be 0.90. This "softer" notion of numeric correctness is useful in scenarios where questions ask for, say, a growth rate or ratio and a model is off by a small factor, or where minor rounding differences occur. We average this score across all questions that require numeric answers. For questions that are not numeric at all (like answers that are text), this isn't directly applicable, so we rely on the ANLS instead.

Using this DeviationScore encourages models to produce answers that are numerically close to correct even if not exact, which is important because sometimes there may be slight ambiguities (for instance, if the answer is 5.0 million, a model might answer 5 million or 5.0, etc.—small differences in formatting or minor rounding shouldn't be overly penalized).

Averaged Normalized Levenshtein Similarity (ANLS): Levenshtein distance between two strings is the minimum number of single-character edits (insertions, deletions, or substitutions) required to change

one string into the other. Normalized Levenshtein similarity (as used in the TextVQA and DocVQA challenges) transforms this distance into a similarity between 0 and 1. Specifically, if $NL(s_1, s_2)$ is the normalized Levenshtein distance between strings s_1 and s_2 (where the distance is divided by the length of the longer string to normalize), then we can define a similarity $s(s_1, s_2) = 1 - NL(s_1, s_2)$ if $NL < \tau$, and 0 otherwise, for some threshold τ. The idea of thresholding is to discount cases where the strings are too different as completely wrong (beyond a certain edit distance, it's probably a different answer entirely).

We adopt the formulation from ICDAR challenges [174], [175]: Suppose each question q has one or more ground-truth answers a_{ij} (where j indexes possible acceptable answers for question i). Let $\widehat{o}_q$ be the predicted answer string. We define:

$$\begin{aligned} \text{ANLS} \;&=\; \frac{1}{N}\sum_{i=1}^{N}\Big(\max_{j}\; s\big(a_{ij}, \widehat{o}_{q_i}\big)\Big), \\ s\big(a_{ij}, \widehat{o}_{q_i}\big) \;&=\; \begin{cases} 1 - NL\big(a_{ij}, \widehat{o}_{q_i}\big), & \text{if } NL\big(a_{ij}, \widehat{o}_{q_i}\big) < \tau, \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{6.2}$$

where N is the number of questions in the evaluation set, and $\tau = 0.5$ is the threshold in our implementation.

What this means is: for each question, compute the normalized Levenshtein similarity between the model's answer and each of the reference answers; take the maximum of those (i.e., assume the reference that best matches the model's output is the intended one); that is the score for that question. Then average these scores over all questions.

We set $\tau = 0.5$ so that if the normalized distance is 50% or more (meaning the model's answer is less than half similar to the true answer in terms of characters), we consider it a complete miss (score 0). If it's above that, we give partial credit linearly. This helps in cases with minor OCR errors. For example, if the correct answer is "Inventory" and the model read it as "Invertory" due to an OCR slip, the edit distance is 1 out of 9 characters (roughly 0.11 normalized), so similarity $s \approx 0.89$ — quite high, meaning the model would get 0.89 credit. But if the model answered "Investment" (completely different meaning), the edit distance is larger relative to length, likely exceeding the 0.5 threshold, giving 0 credit.

For answers that are purely numeric or a mix (like "5 million"), we actually combine both approaches. Specifically, if an answer is numeric, we use the DeviationScore and the ANLS. If an answer is textual, we use ANLS alone. If an answer has both text and number (like a currency "5 million USD"), we evaluate the numeric part with deviation and the text part with ANLS or exact match, etc. But to keep things simpler, in practice, we often either convert everything to a string and use ANLS, or separate

the numeric and text part and score them separately then combine (for example, a weighted average or product).

In EviFiVQA's evaluation script, we handle numeric answers by combining DeviationScore and ANLS via an L2 norm as follows: We compute both the DeviationScore (treating no error as 1, some error downwards) and an ANLS (for the string form). Then define

$$\text{AnswerScore}_{\text{numeric}} = \frac{\sqrt{\left(\text{Deviation Score}\right)^2 + \left(\text{ANLS}\right)^2}}{\sqrt{2}}.$$

This essentially treats DeviationScore and ANLS as two orthogonal components of correctness and finds a combined score (and normalizes to [0,1]). This is a bit heuristic but it worked well in handling cases where, say, the model got the number nearly right but maybe formatted differently.

For answers that are non-numeric, we simply take AnswerScore $= \text{ANLS}$.

To summarize, the **Final Answer Score** for each question is computed as:

- If the ground truth answer is a number or contains a number, combine numeric deviation and ANLS as above.
- If the answer is purely text, use ANLS.
- If the question has multiple correct answers (like any of several synonyms or textual variants are acceptable), we use the maximum ANLS among them (so the model just needs to match one reference well).

This multi-faceted scoring ensures robustness to minor errors. It penalizes big mistakes heavily (score goes to 0 if you're way off numerically or textually), but small mistakes only mildly (score maybe 0.8–0.9 for a tiny typo or a rounding difference).

6.4.3 Combined Metric

Finally, to get a single measure of performance that accounts for both answer accuracy and evidence localization, we define the **Combined Score** for a QA pair as the product of the Answer Score and the Evidence Score. For a given question q, let $\text{AnswerScore}(q)$ be the score (between 0 and 1) obtained from the answer evaluation (as described in Section 6.4.2), and let $\text{EvidenceF1}(q)$ be the evidence localization F1 score for that question (from Section 6.4.1). We then define:

$$\text{TotalScore}(q) = \text{AnswerScore}(q) \times \text{EvidenceF1}(q). \tag{6.3}$$

This multiplicative scheme effectively requires the model to do well on both fronts to get a high score. For example, if a model gets the answer perfectly right ($\text{AnswerScore} = 1$) but fails to provide

correct evidence ($\text{EvidenceF1} = 0$), the TotalScore is 0, reflecting that an answer without evidence is not considered successful in our benchmark. Conversely, if a model highlights all the correct cells but somehow produces an incorrect final answer, that is also a failure (e.g., it identified the right numbers but added them incorrectly). If a model partially identified the evidence and had a nearly correct answer, it will get a partial score.

By using the product rather than, say, a weighted sum, we enforce a strict condition: both evidence and answer must be correct for full credit. One could worry that this might overly penalize models early on (since current models might not be great at evidence selection yet). Indeed, as we'll see in the experiments, many models struggle with evidence, which drags their combined score very low. However, we view this as proper incentive: an AI system that just gives answers without justification is not adequate for financial QA, so we would not consider it successful even if the answers were numerically okay.

In our reporting, we present Answer Score and Evidence Score separately, as well as the Combined Score (the product). This allows analysis of whether a model is better at answering or evidence or both. But the primary leaderboard metric for EviFiVQA is this Combined Score, aligning with the "audit-ready" requirement.

To illustrate, consider a model's output on a question asking for the total liabilities in 2018, and suppose the correct answer is 500 and the evidence should be two cells (current liabilities = 300 and long-term liabilities = 200). If the model answered "500" but only highlighted the cell for current liabilities:

- AnswerScore might be nearly 1 (assuming it exactly got 500).
- EvidenceF1: Precision = 1/1 = 1 (it predicted one cell, which is part of ground truth), Recall = 1/2 = 0.5 (it missed one cell), so F1 0.67.
- Combined = 1 * 0.67 = 0.67. So about 67% credit.

If another model answered "500" and highlighted both cells correctly, it gets 1 * 1 = 1.0. If a model highlighted both cells but answered "520" by mistake, then AnswerScore might be, say, 0.9 (if 520 is within 5% of 500), and EvidenceF1 = 1, so Combined = 0.9. If a model answered completely wrong or gave no evidence, Combined would be 0.

This multiplicative interplay is deliberate. In practice, when comparing models, one might find one model gets a higher Combined Score than another even if its raw Answer Score was slightly lower, because it did much better in Evidence Score, which is a trade-off we are happy to see.

In summary, our evaluation protocol ensures that a model to score highly on EviFiVQA must produce both a correct (or nearly correct) answer and the correct set of supporting cells for that answer. We

hope this protocol will drive research into models that reason in a more interpretable way, rather than treating the problem as a black-box prediction. By examining answer vs. evidence sub-scores, we can also diagnose if a model is guessing answers without proper evidence (which would show up as a high answer score but low evidence score, a scenario we find common with zero-shot models in our experiments).

6.5 Experiments

In this section, we describe the experimental setup for benchmarking current state-of-the-art vision-language models on the EviFiVQA dataset, present the quantitative results, and discuss the findings, including qualitative examples of model outputs. The goal is to assess how well existing models handle the dual challenge of numeric reasoning and evidence localization, and to identify where they struggle the most to guide future improvements.

6.5.1 Experiment Setup

We selected several leading large-scale vision-language models (VLMs) that have demonstrated strong performance on general VQA or document understanding tasks as our baselines: Qwen2-VL [58], LLaVA [168], PixTral [105], Llama-3.2-vision [60], and DeepSeek-VL [59]. These models range in size from 7B to 12B parameters and represent a mix of open-source and proprietary (Qwen2-VL is from Alibaba, etc.) systems. Each of these models is a multimodal extension of a strong language model, with a vision encoder (often based on CLIP or ViT) feeding into a language model that generates answers.

All experiments were conducted on a single machine with an Apple M1 Ultra chip (64 GB unified memory). We used Apple's MLX library [176], [177] for deployment, which provides an efficient runtime for these models on Apple hardware (leveraging the ANE neural engine where possible). While not as powerful as multi-GPU setups, this environment was sufficient for our evaluation after some optimization (we processed images one by one due to memory constraints at 7B+ scale).

We evaluated models in two regimes:

Zero-shot: The models are used as-is, with only prompt engineering to encourage them to output the desired format (answer and evidence). This tests the out-of-the-box capabilities of pre-trained VLMs on our task without any fine-tuning specific to EviFiVQA.

Fine-tuned: We fine-tune the models (when supported by the code and our resources) on a portion of the EviFiVQA training set to see how much their performance can improve with exposure to domain-specific data. Given resource limitations, we did a lightweight fine-tuning: we selected one represen-

tative model as the base (which turned out to be the best zero-shot performer) and fine-tuned it gradually.

For prompt engineering in the zero-shot setting, we crafted a unified instruction prompt for each model in natural language. We found that prompting the model to output a JSON structure was an effective way to get both an answer and a list of bounding boxes. Specifically, we prompted with something like:

"You are a financial analysis assistant. You will be given an image of a financial table and a question. Provide your answer in the following JSON format: "ans": <answer>, "bboxes": [<list of bounding box coordinates of cells used>]. Each bounding box should be a tuple of four numbers (x1,y1,x2,y2) denoting the corners of the cell in the image. Use the table data to find the answer and the exact cells as evidence."

Then for each query we actually input the question after such an instruction, along with the image (depending on the interface, some models like LLaVA accept image + text).

All models were given exactly the same prompt structure and content where possible. The coordinates are relative to the image pixel grid; during evaluation we translate them to match ground truth coordinates.

For fine-tuning, due to resource constraints, we did a two-stage process:

1. We first performed a very light "instruction tuning" using a subset of training data to get the model used to the output format and task. We randomly took 1 QA pair per category per training document (so at most 5 per document) to create a smaller balanced training set (this gave on the order of 40k QA pairs, which is only $\sim$3% of the full train set). We fine-tuned each model on this for 2 epochs with a small learning rate (5×10^{-5}) with a cosine decay schedule and a brief 500-step warmup. We did this for each model to see which baseline improved the most.

2. We then selected the best-performing baseline (after step 1) and fine-tuned it further on the entire training dataset for one epoch to push its performance.

We used Low-Rank Adaptation (LoRA) in 4-bit quantization for fine-tuning to save memory, keeping the vision backbone (the image encoder) frozen and allowing updates only in the projection matrices that fuse vision features into the language model. This significantly reduces memory usage and training time, albeit at some cost to maximum achievable performance. In practice, it allowed us to fine-tune a 7B model on a single machine.

During training, the model's outputs were the JSON text which we parsed to compute loss (treating the entire sequence including answer and coords as the target). We did not use any special loss on coor-

Method	#Params	Deviation Score	ANLS Score	Answer Score	Evidence Score	Combined Score
QWEN2-VL [58]	7B	0.28	0.35	0.32	0.06	0.02
LLaVA [168]	7B	0.25	0.33	0.29	0.04	0.01
Pixtral [105]	12B	0.29	0.36	0.33	0.07	0.02
Llama-3.2-vision [60]	11B	0.30	0.38	0.34	0.07	0.02
DeepSeek-VL [59]	7B	0.33	0.47	0.41	0.09	0.04

Table 6.3: Performance comparison of vision-language models in a zero-shot setting. All scores are averaged for each QA pair across all categories of questions. *Combined Score* multiplies *Answer Score* and *Evidence Score*, reflecting the requirement for both a correct answer and correct evidence.

dinates vs answer differently; it's one sequence loss.

All evaluation metrics described in Section 6.4 were computed on the validation and test sets to monitor progress. We paid particular attention to the Combined Score as our main metric, but also looked at the breakdown of Answer vs Evidence to understand improvements.

After fine-tuning, we report the performance of the best model on the test set. For zero-shot, we report all models on the test set with prompt-only.

In Table 6.3, the zero-shot performance of these vision-language models remains uniformly low across all metrics, emphasizing the difficulty of extracting precise information from complex financial statements with no domain-specific adaptation. Even the best-performing model (DeepSeek-VL) achieves only 0.33 numerical deviation and 0.09 Evidence Score, resulting in a Combined Score of 0.04. Such low accuracy underscores two central challenges. First, identifying the correct numeric or textual answer in highly dense layouts requires specialized reasoning about rows, columns, and multi-year structures that generic pre-training does not provide. Second, the models typically fail to highlight the relevant cells, as indicated by Evidence Scores below 0.10 for all methods. This issue is particularly problematic in a financial context, where each value should be explicitly tied to its source for audit ability and regulatory compliance. The net effect is that zero-shot VLMs struggle to achieve reliable performance, as Combined Scores barely exceed 0.01 for most baselines.

When exposed to only a single QA pair per category in each image, every model sees a notable improvement in both exactness of answers and bounding-box localization (see Table 6.4). DeepSeek-VL, for instance, increases its numerical deviation score from 0.33 to 0.48 and its Evidence Score from 0.09 to 0.28, nearly quadrupling the combined Score to 0.16. We, therefore, choose DeepSeek-VL as the baseline for our detailed experiments. While these numbers remain substantially lower than required for real-world adoption, they illustrate how even limited domain-relevant data can guide the model toward better numeric reasoning and visual grounding in complex tabular layouts. Nonetheless, a numerical deviation score of 0.48 and exact match score of 0.28 still implies that roughly three out of four answers

Method	#Params	Deviation Score	ANLS Score	Answer Score	Evidence Score	Combined Score
QWEN2-VL [58]	7B	0.38	0.50	0.44	0.18	0.08
LLaVA [168]	7B	0.35	0.47	0.41	0.11	0.05
Pixtral [105]	12B	0.40	0.51	0.46	0.19	0.09
Llama-3.2-vision [60]	11B	0.43	0.55	0.49	0.21	0.10
DeepSeek-VL [59]	7B	0.48	0.63	0.56	0.28	0.16

Table 6.4: Performance comparison of vision-language models after fine-tuning on one randomly sampled QA pair from each category for each image for two epochs.

are incorrect, and an Evidence Score around 0.28 means most relevant cells are still overlooked. These findings reinforce the need for continued investigation into domain-specific representation learning, especially given the intricate hierarchical structures, multi-year columns, and specialized calculations found in financial statements. Models that can robustly handle ratio analysis, partial sums across rows, and multi-step numeric derivations are crucial for delivering accurate, audit-ready outputs, and the relatively low post fine-tuning scores confirm that bridging this gap is far from trivial.

Category of Questions	Deviation Score	ANLS Score	Answer Score	Evidence Score	Combined Score
Category 1	0.62	0.82	0.73	0.40	0.29
Category 2	0.54	0.77	0.67	0.34	0.23
Category 3	0.57	0.81	0.70	0.36	0.25
Category 4	0.50	0.72	0.62	0.29	0.18
Category 5	0.39	0.63	0.52	0.23	0.12
Averaged	0.53	0.75	0.65	0.32	0.21

Table 6.5: Approximate performance of DeepSeek-VL on each question category after fine-tuning on the entire training dataset for one epoch. All scores are averaged for each QA pair across each categories of questions individually.

The results in Table 6.5 indicate that after fine-tuning with the entire training dataset for one epoch, Category 1 (extractive questions) is significantly more tractable than the others, with both high ANLS (0.82) and a relatively strong Evidence Score (0.40), reflecting the ease of directly mapping a single cell to the correct answer and bounding box when no multi-step operations are needed. In contrast, Category 2 and Category 3, which require ratio computations and multi-year aggregation, respectively, show moderate performance gains in ANLS and Answer Score but still dip below Category 1 in Exact-Match and bounding-box localization, likely due to the extra step of identifying multiple cells before numerical operations. Category 4, requiring hierarchical aggregation across multiple table segments, sees lower numerical deviation score (0.50), and the Evidence Score declines to 0.29, underlining the difficulty of correctly localizing every component in a more involved computation. Category 5, which combines the aggregation challenge with key-value extraction, is consistently the hardest, posting an numerical deviation of 0.39 and a Combined Score of 0.12. This drop is consistent with the model's

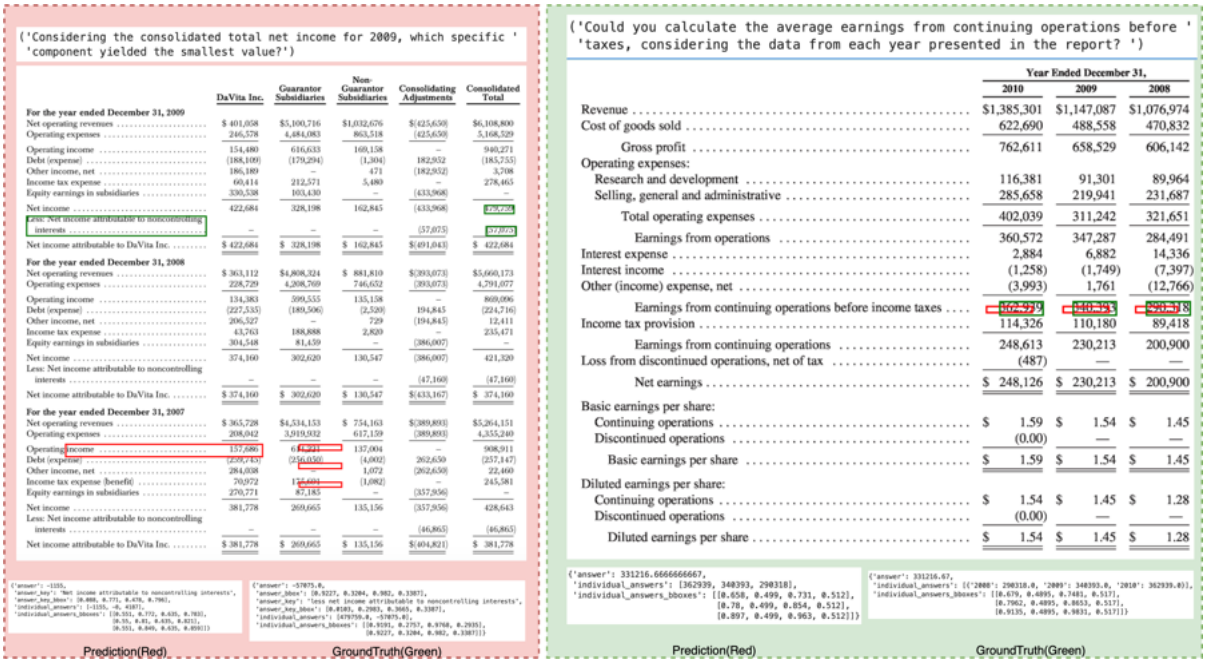


Figure 6.6: Qualitative example from finetuned baseline on 2 different categories of questions from two different samples in the validation set.

struggle to pair the correct numeric figure with its corresponding row header in a multi-hop inference setting. Overall, these category-level results underscore how progressively higher-order reasoning-such as tracking multiple rows and columns or merging hierarchical sub-totals-places an increasing burden on the model's numeric capabilities and ability to ground answers in specific table regions. Figure 6.6 shows two qualitative examples, one negative and one positive for reference.

6.5.2 Attention-Based Evidence Localization

A natural question arising from the VLM baselines is whether cross-modal attention maps—weights between textual query tokens and visual patch tokens—could serve as a proxy for evidence localization, obviating explicit bounding-box generation. This section argues on theoretical and empirical grounds that attention-based localization is architecturally incompatible with the EviFiVQA requirement, and that this incompatibility is structural rather than a consequence of implementation choices. A couple of samples of attention maps visualizations are shown in images 6.7 and 6.8.

Attention Maps Are Not Faithful Explanations The premise underlying attention-based explanation is that high attention weight from a query token to a visual region implies that the model's prediction is causally driven by information in that region. This premise has been experimentally refuted for both LSTM-based [178] and transformer-based [179] architectures: alternative attention distributions can produce identical outputs, and residual connections plus layer normalisation transform attention outputs

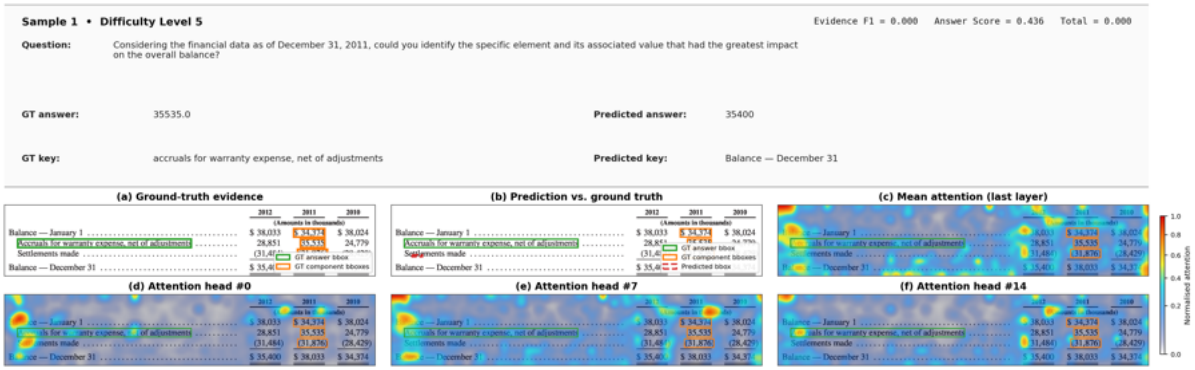


Figure 6.7: Visualization of attention maps on a sample image from EviFiVQA using QWEN2.5 VL 7B model.

before they influence the final prediction, severing any direct causal interpretation. In the EviFiVQA context, even a model that correctly predicts the answer to a Category 3 aggregation question will not necessarily concentrate cross-attention on the relevant cells. The model may attend broadly across the table—allocating weight to row headers, column headers, irrelevant cells, and whitespace—and produce the correct answer by leveraging table structure implicitly encoded in its language model weights. Neither high attention on the correct cells nor low attention is a reliable indicator of whether those cells genuinely drove the prediction.

Spatial Resolution Mismatch ViT-based encoders partition the input image into fixed-size patches, typically 16×16 or 32×32 pixels. At 1024×1024 resolution with a 32×32 patch grid, attention resolution is $32\times32 = 1{,}024$ patches. Table cell boundaries do not align with patch boundaries; a cell spanning pixels (x_1, y_1) to (x_2, y_2) overlaps multiple patches, and the attention distribution mixes evidence from the cell's content, borders, and adjacent cells. Recovering a clean cell-level bounding box from patch attention requires thresholding that introduces hyperparameter sensitivity, and the recovered box cannot be more precise than one patch width. The EviFiVQA evaluation requires IoU ≥ 0.5 between the predicted evidence box and the annotated cell box; for a cell spanning 80 pixels with a 32-pixel patch, the boundary uncertainty alone can preclude achieving this threshold.

Attention Is Not Set-Valued EviFiVQA requires identifying a *set* $G = \{g_1, \ldots, g_{|G|}\}$ of evidence cells, where $|G|$ ranges from 1 to over 20 for Category 4 questions. The Evidence F_1 metric evaluates set-level precision and recall jointly. Attention maps provide a continuous scalar weight per patch, not a discrete set of cell indices. Converting an attention distribution to a set requires thresholding, and the optimal threshold varies by question type, table density, and model state in a way that cannot be determined without access to the ground truth. For hierarchical Category 4 questions, the evidence cells may be spatially dispersed across many rows, sharing no visual feature other than their role in the

Figure 6.8: Visualization of attention maps on a sample image from EviFiVQA using QWEN2.5 VL 7B model.

computation. Attention mechanisms, which encode visual and positional similarity, will not co-activate reliably on such semantically defined but visually heterogeneous evidence sets.

Multi-Head Aggregation Dilutes Evidence Signals Modern transformers compute attention with 8–32 heads per layer across L layers. Aggregating across heads by averaging or maximum projects a high-dimensional attention tensor onto a single spatial map, discarding the specific functional specialization of each head. Rollout methods [180] attempt to propagate attention through all layers, but rely on linearity assumptions that fail for non-linear activations and residual connections. GradCAM applied to ViT patches [181] provides gradient-weighted activations rather than attention weights, but these identify broadly predictive image regions rather than precisely delimited cell boundaries. The empirical consequence is observed directly in Chapter 6: models achieving moderate Answer Scores under zero-shot prompting produce Evidence Scores at or below 0.09, consistent with essentially random spatial evidence generation. This is not a failure of model quality but of the attention mechanism's incompatibility with the cell-set localization task.

The Semantic Gap Between Attention and Arithmetic Reasoning Financial table reasoning requires attending to cells not because of their visual salience but because of their role in a computation. The question "What is the ratio of net revenue in 2007 to 2008?" does not make the 2007 and 2008 net revenue cells visually distinctive—any other year's net revenue is pixelwise identical. The evidence cells are identified by which labels they carry, a function of their position relative to column headers, and by how they participate in the arithmetic operation. Attention mechanisms, encoding visual and positional similarity, cannot distinguish the relevant cell from a visually identical irrelevant cell in an adjacent row or column unless the model has fully encoded the header-cell alignment required to answer the question—and even then this encoding resides in hidden-state representations, not in interpretable attention weights.

The Required Architecture for EviFiVQA Evidence Localization The analysis above collectively requires that reliable evidence localization be implemented through architectures that explicitly represent the discrete, set-valued, cell-indexed nature of the problem. Concretely:

- *Pointer networks* [182] adapted to visual tables produce a distribution over cell indices rather than continuous spatial coordinates, providing structurally appropriate inductive bias.
- *Cell-node cross-attention*, in which each detected cell is treated as a named node (using its bounding box centroid and extent as position encoding) rather than a collection of patches, allows the decoder to select evidence cells by index.
- *Neuro-symbolic decoders* that first select evidence cell indices and then execute a symbolic arithmetic program over the extracted values produce verifiable computation chains rather than autoregressively generated token sequences.

All three approaches require structured intermediate representations of the table—specifically, cell bounding boxes and their content—as produced by the TSR methods of this thesis. This underscores the foundational role of high-fidelity physical TSR as a prerequisite for interpretable and auditable table VQA, and establishes that EviFiVQA demands a qualitatively new model class rather than a quantitative extension of current VLMs.

6.6 Conclusion

In conclusion, EviFiVQA provides a novel and rigorous platform for training and evaluating evidence-grounded reasoning in the financial domain. By focusing on multi-hop numeric computations, hierarchical table structures, and explicit bounding-box localization, our benchmark highlights the unique challenges present in financial statements – especially those requiring audit-ready, transparent answers. Our experiments with leading vision-language models reveal the considerable gap between general-purpose VQA performance and the specialized needs of real-world financial analysis. Even large pre-trained models struggle with the precise reasoning and evidence tracing that EviFiVQA demands, especially in zero-shot settings, and only marginally improve to moderate levels with fine-tuning.

The dataset's scale and diversity encourage models not just to memorize patterns but to genuinely learn to interpret complex documents. The five reasoning categories in EviFiVQA span a spectrum from simple lookups to complex multi-step inferencing, providing a graded measurement of a model's capabilities. Our analysis shows that as question complexity increases, performance drops off substantially, indicating that current models do not yet possess robust higher-order reasoning or the ability to reliably chain together multiple pieces of information from a table.

We hope that EviFiVQA spurs new research directions in interpretable modeling, domain-adaptive training, and advanced numerical reasoning. For instance, future work might explore hybrid models that

combine symbolic computation (for exact arithmetic) with neural perception (for reading the tables), or incorporate explicit tree-structured reasoning to mirror the hierarchical nature of financial statements. Another promising avenue is the development of techniques to enforce consistency between answer and evidence (so that models cannot generate an answer without corresponding evidence or vice versa). The evaluation metrics proposed in this work – particularly the Combined Score – will help track progress on this front by ensuring that improvements in answer accuracy do not come at the expense of evidence quality (and indeed, require improvements in both).

Ultimately, the goal is to pave the way for more transparent and reliable AI systems in finance. An ideal model would not only answer complex financial queries correctly but also provide an *audit trail* for its answers, much like a human analyst citing figures from reports. EviFiVQA takes an important step toward that goal by providing both the means to train such models and the rigorous yardstick to evaluate them. We believe that advancements driven by this benchmark will extend beyond finance to any domain where trustworthy, explainable question answering is required on structured data – exemplifying how AI can be made not just powerful, but also accountable in its decision-making process.

Chapter 7

Conclusion

7.1 Summary of Contributions and Findings

This dissertation investigated robust table understanding from images, ranging from pixel-level parsing of structure to privacy-preserving deployment and evidence-grounded visual reasoning. The work unified three central contributions—accurate Table Structure Recognition (TSR), privacy-preserving models for sensitive domains, and explainable Visual Question Answering (VQA) on tables—into a coherent progression of research that pushes the boundaries of document image analysis.

The first contribution centered on end-to-end physical Table Structure Recognition through the TabStruct family of models. These models viewed TSR as a hybrid of top–down decomposition into cell hypotheses and bottom–up reconstruction of the grid via learned spatial relations. Structural priors such as alignment, continuity, and non-overlap were embedded directly into the learning objective, producing highly accurate and stable layouts across datasets like PubTabNet, SciTSR, and TableBank. The evolution to TabStruct-Net V2 introduced a hierarchical transformer backbone, an anchor-free detection head, and a self-attentive adjacency predictor, all contributing to consistent improvements in fidelity and robustness.

The second major contribution, TabGuard, tackled the emerging need for privacy in document intelligence. It demonstrated that high-fidelity TSR could be achieved without compromising data confidentiality by operating purely on geometric structure. The client-side content masking scheme of TabGuard strips sensitive text while retaining layout cues, allowing server-side models to function on sanitized inputs. Through careful architectural design—comprising the Table Grid Approximator (TGA) and Table Cell Crypt Network (TCCN)—TabGuard achieved near parity with state-of-the-art non-private TSR methods while removing nearly all textual content from transmitted data.

The third contribution focused on grounding reasoning with evidence in financial Table VQA, culminating in the EviFiVQA benchmark. Unlike previous VQA datasets, EviFiVQA requires each answer to be paired with visual evidence, thereby enforcing explainability and auditing. The evaluation proto-

col accounts for both answer accuracy and Evidence F1, penalizing ungrounded predictions. This dual objective exposed weaknesses in current vision–language models, which often produce correct answers without faithful justification, emphasizing the need for interpretable and auditable reasoning in practical systems.

7.2 Discussion of Limitations and Failure Modes

While the proposed methods in this thesis demonstrate strong empirical performance across multiple benchmarks, a deeper analysis reveals several fundamental limitations that stem from architectural assumptions, representational constraints, and methodological design choices. These limitations are not merely edge cases, but rather expose structural weaknesses in how current systems approach table understanding. In particular, they highlight the difficulty of reconciling geometric perception, structural reasoning, and semantic interpretation within a unified framework. In this section, we critically examine these limitations in detail, emphasizing the underlying causes and their broader implications. The goal is not merely to enumerate shortcomings but to trace each limitation to a root architectural or methodological cause, thereby identifying tractable and principled directions for future research. Two additional cross-cutting limitations—the fundamental constraints of current autoregressive VLMs for structured document reasoning, and the absence of calibrated uncertainty in all models considered—are examined separately, as they affect not only this work but the broader trajectory of the field. These threads represent the consequential barriers to deploying the methods of this thesis in high-stakes workflows.

Limitations of TabStruct-Net, TOD-Net, and TSR-Net

- **Anchor design and the coverage–diversity trade-off.** TabStruct-Net operationalizes cell detection through an anchor-based Mask R-CNN whose Region Proposal Network generates candidate regions from a fixed set of scales and aspect ratios. This design encodes a strong inductive prior: cells are assumed to cluster near a handful of canonical shapes. In practice, table cells exhibit aspect ratios spanning several orders of magnitude within a single table; a narrow footnote column adjacent to a wide merged header, or a single-character numeric cell beside a multi-line text entry, both push far outside any finite anchor set. Cells whose true dimensions satisfy

$$\max_{a \in \mathcal{S}} \mathrm{IoU}(b,\, a) < \tau \tag{7.1}$$

  for threshold τ and anchor set $\mathcal{S}$ are systematically suppressed by non-maximum suppression before the GNN stage ever sees them. This failure is not addressable through training alone; it is a representational limitation of the anchor vocabulary. As table domains diversify—multilingual, historical, mobile-captured, or scientifically authored documents—the gap between $\mathcal{S}$ and the true distribution $\mathcal{D}$ of cell dimensions widens monotonically. Anchor-free detection (pursued in TabStruct-Net V2) partially addresses this, but introduces its own budgetary constraint.

- **Soft alignment constraints and structural discretizations.** The Alignment Loss is a pairwise Euclidean regularizer applied over detected cell coordinates within the same row or column. It functions as a Laplacian smoothing energy over boundary positions and provably reduces intra-row coordinate variance by 35–40% compared with a vanilla Mask R-CNN. However, it provides no hard structural guarantee. In tables where ruling lines are entirely absent and inter-cell whitespace is narrower than the detector's positional uncertainty—a regime common in dense numeric financial tables—the alignment gradient is too small relative to the regression loss to enforce coherent grid formation. The loss is fundamentally a continuous relaxation of a discrete constraint ("cells in the same row share a boundary coordinate"), and no continuous relaxation can resolve the combinatorial ambiguity of which set of boundaries constitutes a globally valid grid. Future work should investigate combinatorial loss functions enforced through constraint propagation layers or differentiable Integer Linear Programming surrogates.

- **Quadratic complexity and global consistency in the DGCNN.** Adjacency prediction in the Dynamic Graph Convolutional Network scales as $O(N^2)$ in the number of detected cells N, both in memory (the adjacency matrix) and in the pairwise classification head. For tables with $N > 200$ cells—common in FinTabNet annual reports with dozens of line items across multiple fiscal periods—this becomes a practical inference bottleneck. More fundamentally, pairwise reasoning without a global structural encoding means the model can classify every individual pair (i, j) with high local confidence while producing a globally inconsistent adjacency graph (, predicting a cycle in the row-adjacency relation, which is impossible in any valid table). The model provides no structural validity guarantee, and post-hoc resolution of inconsistencies relies entirely on the heuristics of the XML reconstruction step.

- **TOD-Net and TSR-Net: unidirectional coupling and rectilinear scope.** The two-stage TOD-Net/TSR-Net pipeline introduces a unidirectional dependency: cell localization is optimized independently of structure inference, and no gradient signal can flow from the structure module back to the detector at test time. The continuity and overlap priors in TOD-Net encourage non-overlapping, gap-free cell tilings but conflict with spanning cells, whose correct representation requires that a single region be adjacent to cells in multiple rows or columns simultaneously. TSR-Net's rectilinear adjacency formulation assumes axis-aligned cells and fails on any table with significant perspective distortion, rotation, or curvature—conditions routinely encountered in mobile-captured document images, which the benchmark datasets (SciTSR, FinTabNet) do not include.

Limitations of TabStruct-Net V2

- **Fixed query budget and density sensitivity.** TabStruct-Net V2 replaces anchors with a Deformable DETR backbone that maintains a fixed pool of Q learnable object queries. When $N > Q$ the model is architecturally incapable of detecting all cells; when $N \ll Q$ unused queries must

be suppressed without producing spurious detections. The optimal Q varies with table density in a way that cannot be determined at inference time without prior knowledge of the table. Setting Q conservatively large accommodates dense tables but wastes computation on simple ones; the sensitivity of performance to this hyperparameter is underexplored in the existing evaluation, and production deployment requires either dynamic query budget allocation or a two-pass estimation strategy.

- **HLViT window size and long-range structural coupling.** The Hierarchical Local-Attention Vision Transformer (HLViT) computes self-attention within fixed spatial windows, reducing memory from $O(n^2)$ to $O(n \cdot w^2)$ where w is the window edge length. This gain sacrifices the ability to model long-range dependencies between cells that are spatially distant but structurally coupled: the first and last rows of a tall table share the same column index and must ultimately be assigned consistent logical coordinates, but no feature interaction between them occurs within a single window. Structural coherence across distant cells is accumulated only gradually through successive downsampling stages of HLViT, and information is inevitably compressed at each stage. The window size is a fixed hyperparameter that cannot adapt to table height at inference time, meaning that very tall tables in FinTabNet consistently challenge the model more than their cell complexity alone would predict.

- **Split-and-merge boundary artefacts.** The split-and-merge inference strategy that extends TabStructNet V2 to large tables (Section 4.3.4) relies on sufficient overlap between adjacent splits to reconcile partial structures. A cell that straddles the boundary between two splits is detected independently in both sub-images, and correct merging depends on the overlap region providing enough visual context to disambiguate the cell's extent. Cells at the boundary with unusual spanning configurations—a cell spanning three columns in an otherwise two-column table—are systematically vulnerable to merge errors, and the post-processing heuristic has no access to global table statistics that would help resolve the ambiguity.

Limitations of TabGuard

- **Failure on sparse tables.** TabGuard's Table Grid Approximator (TGA) infers a coarse structural grid from the spatial distribution of masked text contours. In highly sparse tables—pivot tables, summary dashboards, or tables with many structural empty cells—text contours may be absent from entire rows or columns. With no contours to aggregate, the TGA cannot detect the corresponding separator, the dynamic anchor generator produces no cell candidates in that region, and the Table Cell Crypt Network (TCCN) has no region proposal to refine. The 99.92% ground-truth cell coverage holds on the FinTabNet and SciTSR benchmark distributions; on sparser tables from novel domains this coverage can degrade substantially, and the degradation is not graceful—a missed separator propagates to all cells in the corresponding row or column.

- **Fundamental failure on tables with complex cell entities.** TabGuard's content masking algorithm is predicated on a fundamental assumption: the dominant visual entities within each table cell are *text regions*, whose pixel-level bounding boxes can be detected by a lightweight contour-finding step and blacked out to produce the masked image. The TGA then operates on the residual contour layout to infer cell boundaries. This assumption fails categorically when table cells contain non-textual complex entities such as *embedded figures, photographs, bar charts, sparklines, chemical structure diagrams, or logos*. In such cells, the dominant visual content is a raster or vector graphic whose interior is pixel-dense and whose bounding contour is, at best, the full rectangular cell boundary—precisely the structural information that TabGuard is trying to *recover*, not the content it is trying to *mask*. A content masking step that blacks out these graphic regions destroys structural layout cues entirely, because the relationship between the graphic boundary and the actual cell boundary is determined by layout rules (, padding, margin, vertical centering) rather than by contour geometry. The TGA, receiving a masked image in which entire cell interiors are blacked out rather than localised text runs, cannot distinguish a large image-bearing cell from an empty cell from a partially detected spanning cell. The dynamic anchor generator consequently produces anchors of incorrect size, aspect ratio, and position, and the TCCN has no valid region to detect. This is not a quantitative degradation that might be recovered through fine-tuning; it is a categorical breakdown of the entire privacy-preserving inference chain. Tables from scientific publications (containing embedded plots), annual reports with infographic cells, or pharmaceutical documents with molecular diagrams are all affected. Extending TabGuard to such documents requires either a separate modality-aware masking module that identifies and treats graphic regions differently from text regions, or a fundamentally different structural estimation strategy that operates without any valid content signal—, using only the geometric regularities of cell borders or whitespace patterns when content is absent.

- **Privacy model boundary.** TabGuard does not provide formal privacy guarantees equivalent to differential privacy or homomorphic encryption. The threat model assumes a semi-honest server that does not actively attack the masking; a motivated adversary with knowledge of the masking protocol could potentially reconstruct sensitive information from the spatial distribution of contour extents and gaps, particularly if the table follows a known financial reporting template. The residual information leakage from the JSON contour coordinates transmitted to the server has not been formally quantified; future work should bound this leakage using mutual information or Rényi entropy measures. Furthermore, structural metadata itself—the number of rows, column hierarchy depth, or the presence of certain spanning patterns—can be sensitive in competitive financial contexts, and TabGuard treats this metadata as public by design.

- **Domain gap from masked-image-only training.** The TCCN is trained exclusively on masked images. When deployed in scenarios that permit partial masking—, a client masks only cells containing personally identifiable information while leaving structural content visible—the model

encounters a mixed-domain distribution not represented in training. Transfer performance in this setting has not been evaluated and is likely to be degraded relative to the fully masked case.

Limitations of EviFiVQA

- **Arithmetic consistency assumption and annotation noise.** The automated annotation pipeline is predicated on the assumption that financial statements are arithmetically consistent: subtotals equal the sum of their constituent line items within a small numerical tolerance. This assumption fails for rounding discrepancies, restated figures, or values adjusted for discontinued operations. Questions generated from inconsistent tables may have answers that are technically derivable from the table but do not exactly match the annotated evidence because correct computation requires domain knowledge (, knowing that a figure is net of a footnote adjustment appearing on a separate page). This error class, estimated at approximately 0.4% of annotations in the quality assurance study, cannot be eliminated through automated means without richer ontological annotation of financial statement structure.

- **Evidence cardinality conflation.** The Evidence F_1 score treats all evidence cells as equally important: missing the column header that identifies the fiscal year is penalised identically to missing one of several numerical leaf cells that contribute to a summation. In practice, missing a column header renders the answer entirely unverifiable regardless of whether the numerical cells are correctly identified, whereas missing one of five contributing cells is far less consequential. A semantically weighted Evidence F_1 that assigns higher importance to cells determining question interpretation (headers, row labels) than to cells determining numerical values would provide a more diagnostically useful signal and is a natural direction for benchmark refinement.

- **Single-page scope and ecological validity.** EviFiVQA covers only single-page financial tables from 10-K filings. Real-world financial analysis routinely requires reasoning across multiple pages: total assets may appear on page 42 while the contributing segment figures appear in footnotes on pages 67–71. The single-page restriction was a practical necessity for annotation quality and evidence localisation precision, but it limits ecological validity. Models that perform well on EviFiVQA are not guaranteed to handle cross-page evidence chains, cross-document reconciliation, or multi-period aggregation across separately filed statements. Extending EviFiVQA to multi-page and multi-document settings, with correspondingly richer evidence annotation, is the most impactful near-term development for the benchmark.

7.2.1 Autoregressive VLMs for Structured Document Reasoning

The empirical baselines reveal that current autoregressive VLMs—Qwen2-VL, DeepSeek-VL, LLaVA, Pixtral, and LLaMA-3.2-Vision—achieve combined scores below 0.20 on EviFiVQA even after domain-specific fine-tuning. Beyond the empirical results, there are fundamental architectural reasons why this performance ceiling exists independently of model scale.

Exposure bias and error accumulation. Autoregressive VLMs generate outputs token by token according to $P(y_t \mid y_{<t}, x; \theta)$, where x is the visual-linguistic input. Training uses teacher forcing, supplying ground-truth prefixes $y^*_{<t}$; inference conditions on the model's own previous outputs $\hat{y}_{<t}$. The resulting distributional shift is known as *exposure bias*: errors accumulate multiplicatively over the output sequence. For EviFiVQA, outputs comprise an answer value followed by a list of bounding-box coordinate tuples. A single incorrect digit, or a slightly misspecified bounding box, shifts the conditioning context for all subsequent tokens and may cascade into a response that is globally inconsistent with the table's geometry—a constraint the language model cannot exploit in the absence of an explicit structural prior.

Arithmetic global consistency. Autoregressive models produce locally coherent token sequences that need not satisfy global symbolic constraints. For EviFiVQA Category 2 and 3 questions, the answer must equal the arithmetic result of the values in the predicted evidence cells. The model has no mechanism to enforce or verify this post-hoc consistency. Neither arithmetic error (wrong answer, correct cells) nor localization error (correct answer, wrong cells) produces a gradient signal without a symbolic executor embedded in the training loop. This is precisely the failure mode illustrated in the thesis, where a model achieves a high Answer Score but a near-zero Evidence Score.

Absence of calibrated uncertainty. Current VLMs express confidence through the log-probability of the generated sequence, a quantity known to be poorly calibrated for structured outputs: a model may assign high sequence probability to a hallucinated answer because the lexical pattern is frequent in the pre-training corpus, even when the visual evidence is insufficient. Calibration in the formal sense requires

$$P(\text{correct} \mid \hat{p} = p) = p,$$

where $\hat{p}$ is the model's stated confidence. For EviFiVQA this is a doubly-conditioned requirement: the confidence should be jointly predictive of both answer correctness and evidence correctness. Standard temperature scaling applies to classification tasks and does not extend to the structured output setting. Without calibrated uncertainty there is no principled basis for selective automation: a deployed system cannot identify which predictions to surface to a human reviewer and which to accept automatically.

Hallucination under dense visual inputs. Financial tables are informationally dense: hundreds of numerically similar values appear in a visually monotone layout. This density activates multiple hallucination failure modes in VLMs, where the model generates plausible but factually incorrect content. The patch-based ViT encoder cannot reliably distinguish the value in row i, column j from a numerically identical value in row $i + 1$, column j, particularly when both are small numerals at equivalent font sizes.

7.2.2 Future directions.

Hybrid neuro-symbolic pipelines—in which a VLM identifies evidence cells and a deterministic symbolic executor computes the answer from extracted values—offer a principled remedy for arithmetic consistency. For uncertainty quantification, conformal prediction provides distribution-free finite-sample coverage guarantees that could be adapted to the structured output setting without requiring distributional assumptions. For evidence grounding, architectures that produce explicit pointer distributions over cell indices rather than generating bounding-box coordinates autoregressively constitute a structurally more appropriate inductive bias.

7.3 Lessons Learned

Throughout this thesis, several recurring insights emerged. Embedding structure-aware priors proved invaluable for stabilizing learning and producing human-aligned layouts. These priors acted as learned regularizers that improved grid coherence, continuity, and robustness to noise. At a systems level, the progression from modular detection–parsing pipelines to fully integrated differentiable architectures revealed the advantages of allowing structural cues to propagate through the network.

Another key lesson was the importance of moving beyond aggregate metrics such as overall F1 or accuracy. The introduction of evidence-based evaluation in EviFiVQA highlighted how correctness without faithfulness can erode trust. Accurate yet unexplainable models are unsuitable for deployment in high-stakes domains such as finance or healthcare, where users must verify not only *what* the system predicts but also *why*.

7.4 Explainability for TSR and VQA: Toward Selective Automation

Explainability emerged as a unifying theme across TSR and VQA. Human effort can only be reduced meaningfully if systems are able to distinguish between confidently correct outputs and uncertain ones that require verification. In table recognition, uncertainty must be expressed at the level of individual cells and their adjacencies rather than at the global table level. Similarly, in VQA, models must report calibrated confidence for both evidence localization and final answers.

A promising approach involves selective prediction, where systems automatically abstain on uncertain cases. By associating confidence thresholds on both evidence and answer probabilities, the model can decide when to automate and when to defer to human oversight. This notion of coverage–risk optimization directly balances accuracy and productivity. For instance, calibrated uncertainty over evidence cells allows the system to flag ambiguous areas for manual correction while confidently accepting reliable predictions elsewhere.

For TSR, uncertainty estimation through techniques like deep ensembles or Monte Carlo dropout highlights visually ambiguous or unseen layouts. These can be complemented with structural consistency checks that identify geometric anomalies, ensuring interpretable alerts. In VQA, enforcing evidence-grounded answers ensures that correctness is always supported by visually verifiable cues. The EviFiVQA dataset, by requiring this coupling of evidence and answers, makes explainability an intrinsic part of the evaluation process.

7.5 On Autoregressive VLMs versus Specialized Models

The recent dominance of large autoregressive vision–language models (VLMs) brings both opportunities and challenges for table understanding and VQA. These models exhibit remarkable generalization capabilities but also introduce issues of hallucination, overconfidence, and weak grounding. Because their generation process is autoregressive, exposure bias accumulates over long outputs, often producing coherent but incorrect answers. Moreover, the computational cost and data requirements of large VLMs make them unsuitable for privacy-sensitive or real-time applications.

A more prudent approach for document understanding is to develop specialized, structure-aware models that are lightweight, interpretable, and easier to calibrate. Specialized TSR architectures, with explicit structural priors and adjacency reasoning, provide modular outputs that can be validated independently. For VQA, hybrid pipelines that combine TSR perception with grounded symbolic reasoning offer a natural path toward faithfulness. By decomposing the reasoning process into perception, evidence selection, and constrained computation, such systems minimize hallucination and provide actionable uncertainty estimates. Confidence-calibrated abstention and explicit verification modules further ensure that incorrect outputs are caught early, allowing selective human intervention.

7.6 Privacy Preservation for Tables at Scale

TabGuard demonstrated that structural cues alone can drive effective table understanding, proving that privacy and performance are not mutually exclusive. However, achieving this balance at scale introduces new challenges. As document analysis becomes increasingly decentralized and multi-institutional, future work must develop formal threat models quantifying residual information leakage from masked layouts. Privacy metrics analogous to ROC curves can characterize the trade-off between structural fidelity and potential exposure.

Scalable privacy-preserving document AI will require synergy between geometry-based anonymization, trusted hardware enclaves, homomorphic encryption for numeric computations, and federated training for domain adaptation. Pretraining on large synthetically masked datasets can further amortize the cost of privacy-preserving training and improve generalization to unseen domains. Extending

the privacy paradigm beyond TSR to table-based reasoning and evidence-level VQA, where only minimal coordinates and verified numeric aggregates are shared, will be crucial for practical adoption in finance, healthcare, and enterprise document workflows.

7.7 Evolution of Research in Table Understanding

Table detection and structure recognition have evolved from heuristic line detection methods to deep models leveraging graph networks and transformers. The field now emphasizes structural fidelity through metrics such as TEDS and GriTS. Benchmarks like PubTables-1M, TableBank, and FinTabNet have enabled large-scale supervised training. The TabStruct models presented in this thesis build upon these foundations while introducing inductive biases tailored to table geometry.

Table and document VQA research spans symbolic reasoning approaches, differentiable aggregators, and generative OCR-free systems such as Donut and Pix2Struct. However, many of these models prioritize answer accuracy over grounding, leading to unfaithful reasoning. The EviFiVQA benchmark advances this field by integrating answer and evidence evaluation, encouraging explainable model behavior.

Large vision–language models such as PaLI-X, LLaVA, and Phi-3 Vision demonstrate impressive generalization but lack interpretability and efficiency. Their reliance on massive pretraining data limits their applicability to privacy-sensitive or domain-specific contexts. Specialized, structure-aware models remain preferable for applications demanding transparency and accountability.

Explainability, uncertainty, and selective prediction have become key research themes in trustworthy AI. The integration of calibrated confidence with evidence grounding forms a robust basis for reliable automation. This principle aligns with human-in-the-loop paradigms, where systems assist rather than replace human judgment.

Privacy-preserving document intelligence remains an emerging but vital direction. Traditional text-only obfuscation methods are insufficient for layout-rich documents. Geometry-based masking and client-side processing, as realized in TabGuard, represent a significant step toward scalable privacy without sacrificing structural understanding.

7.8 Concluding Remarks

This thesis advanced the state of document intelligence by unifying structural accuracy, privacy, and explainability. From TabStruct's geometry-driven parsing to TabGuard's privacy-preserving design and EviFiVQA's evidence-grounded reasoning, the contributions collectively demonstrate that reliable

automation requires models that are not only accurate but also interpretable and secure. The future of document AI lies in systems that know when to act and when to defer, that can justify every decision, and that protect the privacy of the documents they serve. Extending these principles to other structured document modalities such as forms, charts, and hybrid layouts will be a natural continuation of this work.

Bibliography

[1] Sachin Raja, Ajoy Mondal, and C. V. Jawahar, "Table structure recognition using top-down and bottom-up cues," in *European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 70–86.

[2] Sachin Raja, Ajoy Mondal, and CV Jawahar, "Visual understanding of complex table structures from document images," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2299–2308.

[3] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes, "Image-based table recognition: Data, model, and evaluation," in *European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 564–580.

[4] Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang, "Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 697–706.

[5] Azka Gilani, Shah Rukh Qasim, Imran Malik, and Faisal Shafait, "Table detection using deep learning," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, vol. 1, 2017, pp. 771–776.

[6] Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang, "TableSense: Spreadsheet table detection with convolutional neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[7] Isaak Kavasidis, C Pino, S Palazzo, *et al.*, "A saliency-based convolutional neural network for table and chart detection in digitized documents," in *Italian Conference on Image Analysis and Processing (ICIAP)*, 2019.

[8] Jianying Hu, Ramanujan S Kashi, Daniel P Lopresti, and Gordon Wilfong, "Medium-independent table detection," in *Document Recognition and Retrieval VII*, 1999.

[9] Yalin Wang, Ihsin T Phillips, and Robert M Haralick, "Table structure understanding and its performance evaluation," *Pattern Recognition*, 2004.

[10] Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C. Lee Giles, "Learning to extract semantic structure from documents using multimodal fully convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 5315–5324.

[11] Dario Augusto Borges Oliveira and Matheus Palhares Viana, "Fast CNN-based document layout analysis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, IEEE, 2017, pp. 1–9.

[12] Shah Rukh Qasim, Hassan Mahmood, and Faisal Shafait, "Rethinking table parsing using graph neural networks," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[13] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li, "TableBank: Table benchmark for image-based table detection and recognition," in *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, European Language Resources Association, 2020, pp. 1918–1925.

[14] Panupong Pasupat and Percy Liang, "Compositional semantic parsing on semi-structured tables," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics, 2015, pp. 1470–1480. DOI: 10.3115/v1/P15-1142.

[15] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, *et al.*, "Natural questions: A benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[16] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 4320–4333. DOI: 10.18653/v1/2020.acl-main.398.

[17] Julian Martin Eisenschlos, Syrine Krichene, and Thomas Müller, "Understanding tables with intermediate pre-training," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Association for Computational Linguistics, 2020, pp. 281–296. DOI: 10.18653/v1/2020.findings-emnlp.27.

[18] Christopher Tensmeyer, Vlad Morariu, Brian Price, Scott Cohen, and Tony Martinezp, "Deep splitting and merging for table structure decomposition," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[19] Thomas G Kieninger, "Table structure recognition based on robust block segmentation," in *Document Recognition V*, 1998.

[20] Katsuhiko Itonori, "Table structure recognition based on textblock arrangement and ruled line position," in *International Conference on Document Analysis and Recognition (ICDAR)*, 1993.

[21] E Green and M Krishnamoorthy, "Recognition of tables using table grammars," in *Annual Symposium on Document Analysis and Information Retrieval*, 1995.

[22] Rujiao Long, Wen Wang, Nan Xue, *et al.*, "Parsing table structures in the wild," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 944–952.

[23] Alon Talmor, Ori Yoran, Amnon Catav, *et al.*, "Multimodalqa: Complex question answering over text, tables and images," in *International Conference on Learning Representations*, 2021. [Online]. Available: `https://openreview.net/forum?id=ee6W5UgQLa`.

[24] Zewen Chi, Heyan Huang, Heng-Da Xu, Houjin Yu, Wanxuan Yin, and Xian-Ling Mao, *Complicated table structure recognition*, arXiv:1908.04729, Preprint, 2019.

[25] Scott Tupaj, Zhongwen Shi, C Hwa Chang, and Hassan Alam, "Extracting tabular information from text files," *EECS Department, Tufts University, Medford, USA*, 1996.

[26] Huawen Shen, Xiang Gao, Jin Wei, *et al.*, "Divide rows and conquer cells: Towards structure recognition for large tables," IJCAI, 2023.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] Ze Liu, Yutong Lin, Yue Cao, *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[29] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask R-CNN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, Curran Associates, Inc., 2015.

[31] Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, and Yoshihiro Matsuo, "Understanding the semantic structures of tables with a hybrid deep neural network architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[32] Sachin Raja, Ajoy Mondal, and CV Jawahar, "Evifivqa: A benchmark for evidence-grounded multi-hop reasoning in financial vqa," in *International Conference on Document Analysis and Recognition*, Springer, 2025, pp. 587–604.

[33] Sachin Raja, Ajoy Mondal, and CV Jawahar, "Icdar 2023 competition on visual question answering on business document images," in *International Conference on Document Analysis and Recognition*, Springer, 2023, pp. 454–470.

[34] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, *et al.*, "Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, Association for Computational Linguistics, 2021, pp. 3277–3287. arXiv: 2105.07624. [Online]. Available: https://aclanthology.org/2021.acl-long.254/.

[35] Zhiyu Chen, Wenhu Chen, Charese Smiley, *et al.*, "Finqa: A dataset of numerical reasoning over financial data," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2021, pp. 3697–3711. arXiv: 2109.00122. [Online]. Available: https://aclanthology.org/2021.emnlp-main.300/.

[36] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar, "Docvqa: A dataset for vqa on document images," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 2200–2209.

[37] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, *et al.*, "TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, Association for Computational Linguistics, 2021, pp. 3277–3287. DOI: 10.18653/v1/2021.acl-long.254.

[38] Martin Holeček, Antonín Hoskovec, Petr Baudiš, and Pavel Klinger, *Line-items and table understanding in structured documents*, arXiv:1910.11573, Preprint, 2019.

[39] Pau Riba, Anjan Dutta, Lutz Goldmann, Alicia Fornes, Oriol Ramos, and Josep Llados, "Table detection in invoice documents by graph neural networks," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[40] Harsh Desai, Pratik Kayal, and Mayank Singh, "Tablex: A benchmark dataset for structure and content information extraction from scientific tables," in *Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II 16*, Springer, 2021, pp. 554–569.

[41] Liangcai Gao, Yilun Huang, Hervé Déjean, *et al.*, "ICDAR 2019 competition on table detection and recognition (cTDaR)," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[42] Brandon Smock, Rohith Pesala, and Robin Abraham, "GriTS: Grid table similarity metric for table structure recognition," in *Document Analysis and Recognition – ICDAR 2023: 17th International Conference, San José, CA, USA, August 21–26, 2023, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 14193, Springer, 2023, pp. 105–122.

[43] Tarun Kumar and Himanshu Sharad Bhatt, *Evaluating table structure recognition: A new perspective*, arXiv:2208.00385, Preprint, 2022.

[44] Bin Xiao, Murat Simsek, Burak Kantarci, and Ala Abu Alkheir, *Table structure recognition with conditional attention*, arXiv:2203.03819, Preprint, 2022.

[45] Ray Smith, "An overview of the Tesseract OCR engine," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2007.

[46] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed, "DeepDeSRT: Deep learning for detection and structure recognition of tables in document images," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, vol. 1, 2017, pp. 1162–1167.

[47] Sachin Raja, Ajoy Mondal, and C. V. Jawahar, "Treading towards privacy-preserving table structure recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025.

[48] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi, "ICDAR 2013 table competition," in *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2013, pp. 1449–1453.

[49] Asif Shahab, Faisal Shafait, Thomas Kieninger, and Andreas Dengel, "An open approach towards the benchmarking of table structure recognition systems," in *IAPR International Workshop on Document Analysis Systems (DAS)*, 2010.

[50] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes, "Image-based table recognition: Data, model, and evaluation," in *European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 564–580.

[51] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li, "TableBank: Table benchmark for image-based table detection and recognition," in *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, European Language Resources Association, 2020, pp. 1918–1925.

[52] Sachin Raja, Ajoy Mandal, and CV Jawahar, "Treading towards privacy-preserving table structure recognition," in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2025, pp. 2311–2321.

[53] Sachin Raja, Ajoy Mondal, and CV Jawahar, "From pixels to tables: Reconstructing complex tables from document images," *International Journal on Document Analysis and Recognition (IJDAR)*, pp. 1–16, 2025.

[54] Basilios Gatos, Dimitrios Danatsas, Ioannis Pratikakis, and Stavros J. Perantonis, "Automatic table detection in document images," in *IAPR International Workshop on Document Analysis Systems (DAS)*, IAPR, 2005.

[55] Li Deng, Shuo Zhang, and Krisztian Balog, "Table2Vec: Neural word and entity embeddings for table population and retrieval," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2019.

[56] Wenyuan Xue, Baosheng Yu, Wen Wang, Dacheng Tao, and Qingyong Li, "Tgrnet: A table graph reconstruction network for table structure recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1295–1304.

[57] Avinash Anand, Raj Jaiswal, Pijush Bhuyan, *et al.*, "TC-OCR: TableCraft OCR for efficient detection and recognition of table structure and content," in *Proceedings of the ACM Symposium on Document Engineering (DocEng)*, ACM, 2023, pp. 11–18.

[58] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, *et al.*, *Qwen2-VL: Enhancing vision-language modelś perception of the world at any resolution*, arXiv:2409.12191, Preprint, 2024.

[59] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, *et al.*, *DeepSeek-VL: Towards real-world vision-language understanding*, arXiv:2403.05525, Preprint, 2024.

[60] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, *et al.*, *The Llama 3 herd of models*, arXiv:2407.21783, Preprint, 2024.

[61] Thotreingam Kasar, Philippine Barlas, Sébastien Adam, Clément Chatelain, and Thierry Paquet, "Learning to detect tables in scanned document images using line information," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2013.

[62] Ana Costa e Silva, "Learning rich hidden Markov models in document analysis: Table location," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2009.

[63] MAC Akmal Jahan and Roshan G Ragel, "Locating tables in scanned documents for reconstructing and republishing," in *International Conference on Intelligent and Advanced Systems (ICIAS)*, 2014.

[64] Brandon Smock, Rohith Pesala, and Robin Abraham, *Aligning benchmark datasets for table structure recognition*, arXiv:2303.00716, Preprint, 2023.

[65] Brandon Smock, Rohith Pesala, and Robin Abraham, "Pubtables-1m: Towards comprehensive table extraction from unstructured documents," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4634–4642.

[66] Pau Riba, Lutz Goldmann, Oriol Ramos Terrades, Diede Rusticus, Alicia Fornés, and Josep Lladós, "Table detection in business document images by message passing networks," *Pattern Recognition*, vol. 127, p. 108 641, 2022.

[67] Liang Qiao, Zaisheng Li, Zhanzhan Cheng, *et al.*, "LGPMA: Complicated table structure recognition with local and global pyramid mask alignment," in *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part I*, Springer, 2021, pp. 99–114.

[68] Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar, "Tableformer: Table structure understanding with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4614–4623.

[69] Weihong Lin, Zheng Sun, Chixiang Ma, *et al.*, "TSRFormer: Table structure recognition with transformers," in *Proceedings of the 30th ACM International Conference on Multimedia*, Association for Computing Machinery, 2022, pp. 3981–3990. DOI: 10.1145/3503161.3548038.

[70] Hao Liu, Xin Li, Bing Liu, Deqiang Jiang, Yinsong Liu, and Bo Ren, "Neural Collaborative Graph Machines for table structure recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 4533–4542.

[71] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang, "Webtables: Exploring the power of tables on the web," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 538–549, 2008.

[72] Varish Mulwad, Tim Finin, and Anupam Joshi, "Semantic message passing for generating linked data from tables," in *International Semantic Web Conference*, Springer, 2013, pp. 363–378.

[73] Junwei Bao, Duyu Tang, Nan Duan, *et al.*, "Table-to-text: Describing table region with natural language," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[74] Shuo Zhang and Krisztian Balog, "Web table extraction, retrieval, and augmentation: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 2, pp. 1–35, 2020.

[75] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu, "Turl: Table understanding through representation learning," *ACM SIGMOD Record*, vol. 51, no. 1, pp. 33–40, 2022.

[76] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Association for Computational Linguistics, 2002, pp. 311–318.

[77] Chin-Yew Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, Association for Computational Linguistics, 2004, pp. 74–81.

[78] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh, "CIDEr: Consensus-based image description evaluation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015, pp. 4566–4575.

[79] Panupong Pasupat and Percy Liang, "Compositional semantic parsing on semi-structured tables," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP), Volume 1: Long Papers*, Association for Computational Linguistics, 2015, pp. 1470–1480. DOI: 10.3115/v1/P15-1142.

[80] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 8413–8426. DOI: 10.18653/v1/2020.acl-main.745. arXiv: 2005.08314. [Online]. Available: https://aclanthology.org/2020.acl-main.745/.

[81] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou, "Layoutlm: Pre-training of text and layout for document image understanding," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1192–1200.

[82] Yang Xu, Yiheng Xu, Tengchao Lv, *et al.*, "LayoutLMv2: Multi-modal pre-training for visually-rich document understanding," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics, 2021, pp. 2579–2591. DOI: 10.18653/v1/2021.acl-long.201.

[83] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei, "Layoutlmv3: Pre-training for document ai with unified text and image masking," in *Proceedings of the 30th ACM International Conference on Multimedia (MM)*, Association for Computing Machinery, 2022, pp. 4083–4091. DOI: 10.1145/3503161.3548112. [Online]. Available: https://arxiv.org/abs/2204.08387.

[84] Jiapeng Wang, Lianwen Jin, and Kai Ding, "LiLT: A simple yet effective language-independent layout transformer for structured document understanding," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2022, pp. 7747–7757. DOI: 10.18653/v1/2022.acl-long.534.

[85] Junlong Li, Yiheng Xu, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei, "DiT: Self-supervised pre-training for document image transformer," in *Proceedings of the 30th ACM International Conference on Multimedia*, Association for Computing Machinery, 2022, pp. 3530–3539. DOI: 10.1145/3503161.3547911.

[86] Alec Radford, Jong Wook Kim, Chris Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PmLR, 2021, pp. 8748–8763.

[87] Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao, "Neural enquirer: Learning to query tables in natural language," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, IJCAI/AAAI Press, 2016, pp. 2308–2314.

[88] Xiaojun Xu, Chang Liu, and Dawn Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," in *Proceedings of the VLDB Endowment (PVLDB)*, vol. 11, 2018, pp. 1340–1353. arXiv: 1711.04436. [Online]. Available: https://arxiv.org/abs/1711.04436.

[89] Bailin Wang, Richard Shin, Xiaodong Liu, *et al.*, "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 7567–7578. arXiv: 1911.04942. [Online]. Available: https://aclanthology.org/2020.acl-main.677.pdf.

[90] Minesh Mathew, Viraj Bagal, Rubén Pérez Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar, "Infographicvqa," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, pp. 1697–1706. arXiv: 2104.12756. [Online]. Available: https://openaccess.thecvf.com/content/WACV2022/html/Mathew_InfographicVQA_WACV_2022_paper.html.

[91] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque, "ChartQA: A benchmark for question answering about charts with visual and logical reasoning," in *Findings of the Association for Computational Linguistics: ACL 2022*, Association for Computational Linguistics, 2022, pp. 2263–2279. DOI: 10.18653/v1/2022.findings-acl.177.

[92] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes, "Image-based table recognition: Data, model, and evaluation," in *Proceedings of the 16th International Conference on Document Analysis and Recognition (ICDAR)*, PubTabNet and TEDS, IEEE, 2021, pp. 144–153. [Online]. Available: https://arxiv.org/abs/1911.10683.

[93] Rafał Powalski, Łukasz Borchmann, Dawid Jurkiewicz, Tomasz Dwojak, Michał Pietruszka, and Gabriela Pałka, "Going full-tilt boogie on document understanding with text-image-layout transformer," in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM)*, Association for Computing Machinery, 2021, pp. 3143–3152. DOI: 10.1145/3459637.3482013. [Online]. Available: https://arxiv.org/abs/2102.09550.

[94] Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha, "Docformer: End-to-end transformer for document understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 993–1003. arXiv: 2106.11539. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/papers/Appalaraju_DocFormer_End-to-End_Transformer_for_Document_Understanding_ICCV_2021_paper.pdf.

[95] Geewook Kim, Teakgyu Hong, Moonbin Yim, *et al.*, "Donut: Document understanding transformer without ocr," in *European Conference on Computer Vision (ECCV)*, Springer, 2022.

DOI: 10.1007/978-3-031-19815-1_29. arXiv: 2111.15664. [Online]. Available: https://arxiv.org/abs/2111.15664.

[96] Kenton Lee, Mandar Joshi, Iulia Turc, *et al.*, "Pix2struct: Screenshot parsing as pretraining for visual language understanding," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, PMLR, 2023, pp. 18 851–18 869. arXiv: 2210.03347. [Online]. Available: https://proceedings.mlr.press/v202/lee23g/lee23g.pdf.

[97] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi, "Mathqa: Towards interpretable math word problem solving with operation-based formalisms," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, 2019, pp. 2357–2367. DOI: 10.18653/v1/N19-1245.

[98] Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang, "Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2022, pp. 6588–6600. DOI: 10.18653/v1/2022.acl-long.454.

[99] Xi Chen, Josip Djolonga, *et al.*, *Pali-x: On scaling up a multilingual vision and language model*, arXiv:2305.18565, Preprint, 2023.

[100] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee, "Visual instruction tuning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, Curran Associates, Inc., 2023, pp. 34 892–34 916.

[101] Marib Abdin, Jyoti Aneja, H. Awadalla, Ahmed Awadallah, *et al.*, *Phi-3 technical report: A highly capable language model locally on your phone*, arXiv:2404.14219, Preprint, 2024.

[102] Jiapeng Li, Yiheng Xu, Tengchao Lv, *et al.*, "Dit: Self-supervised pre-training for document image transformer," in *Proceedings of the 30th ACM International Conference on Multimedia (MM)*, Association for Computing Machinery, 2023, pp. 1150–1160. DOI: 10.1145/3503161.3547911. [Online]. Available: https://arxiv.org/abs/2203.02378.

[103] Jiabo Ye, Anwen Hu, Haiyang Xu, *et al.*, "UReader: Universal OCR-free visually-situated language understanding with multimodal large language model," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, Association for Computational Linguistics, 2023, pp. 7861–7875.

[104] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[105] Pravesh Agrawal, Szymon Antoniak, *et al.*, *Pixtral 12b*, arXiv:2410.07073, Preprint, 2024.

[106] Michał Turski, Tomasz Stanisławek, Karol Kaczmarek, Paweł Dyda, and Filip Graliński, "Ccpdf: Building a high quality corpus for visually rich documents from web crawl data," in *International Conference on Document Analysis and Recognition*, Springer, 2023, pp. 348–365.

[107] Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S Nassar, and Peter Staar, "Doclaynet: A large human-annotated dataset for document-layout segmentation," in *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2022, pp. 3743–3751.

[108] Mindee, *Doctr: Document text recognition*, `https://github.com/mindee/doctr`, 2021.

[109] Jing Fang, Xin Tao, Zhi Tang, Ruiheng Qiu, and Ying Liu, "Dataset, ground-truth and performance metrics for table detection evaluation," in *2012 10th IAPR International Workshop on Document Analysis Systems*, IEEE, 2012, pp. 445–449.

[110] Zewen Chi, Heyan Huang, Heng-Da Xu, Houjin Yu, Wanxuan Yin, and Xian-Ling Mao, "Complicated table structure recognition," in *Proceedings of the 28th International Conference on Computational Linguistics (COLING) Workshops*, Introduces the SciTSR dataset, 2020. [Online]. Available: `https://arxiv.org/abs/1908.04729`.

[111] Tao Yu, Rui Zhang, Kai Yang, *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2018, pp. 3911–3921.

[112] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang, "Search-based neural structured learning for sequential question answering," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1821–1831.

[113] Ahmad Al Badawi, Chao Jin, Jie Lin, *et al.*, "Towards the alexnet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1330–1343, 2020.

[114] Kuan-Yu Chu, Yin-Hsi Kuo, and Winston H. Hsu, "Real-time privacy-preserving moving object detection in the cloud," in *Proceedings of the 21st ACM International Conference on Multimedia*, 2013, pp. 597–600.

[115] Ryo Yonetani, Vishnu Naresh Boddeti, Kris M. Kitani, and Yoichi Sato, "Privacy-preserving visual learning using doubly permuted homomorphic encryption," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2040–2050.

[116] Keonhyeong Kim and Im Young Jung, "Secure object detection based on deep learning," *Journal of Information Processing Systems*, vol. 17, no. 3, pp. 571–585, 2021.

[117] Yang Liu, Anbu Huang, Yun Luo, *et al.*, "Fedvision: An online visual object detection platform powered by federated learning," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020, pp. 13 172–13 179.

[118] Yang Liu, Anbu Huang, Yun Luo, *et al.*, "Federated learning-powered visual object detection for safety monitoring," *AI Magazine*, vol. 42, no. 2, pp. 19–27, 2021.

[119] Peihua Yu and Yunfeng Liu, "Federated object detection: Optimizing object detection model with federated learning," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, 2019, pp. 1–6.

[120] Yunlong Mao, Shanhe Yi, Qun Li, Jinghao Feng, Fengyuan Xu, and Sheng Zhong, "Learning from differentially private neural activations with edge computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018, pp. 90–102.

[121] Aditya Golatkar, Alessandro Achille, Yu-Xiang Wang, Aaron Roth, Michael Kearns, and Stefano Soatto, "Mixed differential privacy in computer vision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8376–8386.

[122] Zhiqi Bu, Jialin Mao, and Shiyun Xu, "Scalable and efficient training of large convolutional neural networks with differential privacy," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 38 305–38 318.

[123] Baoping Wang, Duanyang Feng, Junyu Su, and Shiyang Song, "An effective federated object detection framework with dynamic differential privacy," *Mathematics*, vol. 12, no. 14, p. 2150, 2024.

[124] Imen Chakroun, Tom Vander Aa, Roel Wuyts, and Wilfried Verachtert, "Privacy-preserving multi-party machine learning for object detection," in *2021 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*, 2021, pp. 7–13.

[125] Tianyu Bai, Song Fu, and Qing Yang, "Privacy-preserving object detection with secure convolutional neural networks for vehicular edge computing," *Future Internet*, vol. 14, no. 11, p. 316, 2022.

[126] Ruonan Wang, Min Luo, Qi Feng, Cong Peng, and Debiao He, "Multi-party privacy-preserving faster R-CNN framework for object detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2023, to appear.

[127] Andrew G. Howard, Menglong Zhu, Bo Chen, *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[128] Ayesha Younis, Shixin Li, J. N. Shelembi, and Hai Zhang, "Real-time object detection using pre-trained deep learning models mobilenet-SSD," in *Proceedings of the 6th International Conference on Computing and Data Engineering (ICCDE)*, 2020, pp. 44–48.

[129] Yu-Chen Chiu, Chi-Yi Tsai, Mind-Da Ruan, Guan-Yu Shen, and Tsu-Tian Lee, "MobileNet-SSDv2: An improved object detection model for embedded systems," in *2020 International Conference on System Science and Engineering (ICSSE)*, 2020, pp. 1–5.

[130] X. Yi, L. Gao, Y. Liao, X. Zhang, R. Liu, and Z. Jiang, "CNN based page object detection in document images," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[131] Shubham Singh Paliwal, D Vishwanath, Rohit Rahul, Monika Sharma, and Lovekesh Vig, "TableNet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[132] Saqib Ali Khan, Syed Muhammad Daniyal Khalid, Muhammad Ali Shahzad, and Faisal Shafait, "Table structure extraction with Bi-directional Gated Recurrent Unit networks," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[133] Shoaib Ahmed Siddiqui, Pervaiz Iqbal Khan, Andreas Dengel, and Sheraz Ahmed, "Rethinking semantic segmentation for table structure recognition in documents," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 1397–1402. DOI: 10.1109/icdar.2019.00225.

[134] Wenyuan Xue, Qingyong Li, and Dacheng Tao, "ReS2TIM: Reconstruct syntactic structures from table images," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[135] Ajoy Mondal, Peter Lipps, and C V Jawahar, "IIIT-AR-13K: a new dataset for graphical object detection in documents," in *IAPR International Workshop on Document Analysis Systems (DAS)*, 2020.

[136] Ranajit Saha, Ajoy Mondal, and C. V. Jawahar, "Graphical object detection in document images," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[137] Asif Shahab, Faisal Shafait, Thomas Kieninger, and Andreas Dengel, "An open approach towards the benchmarking of table structure recognition systems," in *IAPR International Workshop on Document Analysis Systems (DAS)*, 2010.

[138] Ross Girshick, "Fast R-CNN," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, 2015, pp. 1440–1448.

[139] Fisher Yu and Vladlen Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016.

[140] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 4, pp. 834–848, 2018.

[141] Sanghyun Woo, Soonmin Hwang, Ho-Deok Jang, and In So Kweon, "Gated bidirectional feature pyramid network for accurate one-shot detection," *Machine Vision and Applications*, 2019.

[142] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[143] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Transactions on Graphics*, vol. 38, no. 5, pp. 1–12, 2019.

[144] Hangdi Xing, Feiyu Gao, Rujiao Long, *et al.*, "LORE: Logical location regression network for table structure recognition," in *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI)*, AAAI Press, 2024, pp. 6185–6193.

[145] Hao Liu, Xin Li, Mingming Gong, *et al.*, *Grab what you need: Rethinking complex table structure recognition with flexible components deliberation*, arXiv:2303.09174, Preprint, 2023.

[146] Xuran Pan, Tianzhu Ye, Zhuofan Xia, Shiji Song, and Gao Huang, "Slide-transformer: Hierarchical vision transformer with local self-attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 2082–2091.

[147] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," in *International Conference on Learning Representations*, 2021. [Online]. Available: `https://openreview.net/forum?id=gZ9hCDWe6ke`.

[148] Shilong Liu, Tianhe Ren, Jiayu Chen, *et al.*, "Detection transformer with stable matching," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6491–6500.

[149] Yongshuai Huang, Ning Lu, Dapeng Chen, *et al.*, "Improving table structure recognition with visual-alignment sequential coordinate modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 134–11 143.

[150] Pengyuan Lyu, Weihong Ma, Hongyi Wang, *et al.*, "Gridformer: Towards accurate table structure recognition via grid prediction," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 7747–7757.

[151] Qiyu Hou and Jun Wang, "Tablet: Table structure recognition using encoder-only transformers," in *International Conference on Document Analysis and Recognition*, Springer, 2025, pp. 253–278.

[152] Yitong Zhou, Mingyue Cheng, Qingyang Mao, Feiyang Xu, and Xin Li, "Enhancing table recognition with vision llms: A benchmark and neighbor-guided toolchain reasoner," *arXiv preprint arXiv:2412.20662*, 2024.

[153] Yitong Zhou, Mingyue Cheng, Qingyang Mao, *et al.*, "Benchmarking multimodal llms on recognition and understanding over chemical tables," *arXiv preprint arXiv:2506.11375*, 2025.

[154] Binyuan Hui, Jian Yang, Zeyu Cui, *et al.*, "Qwen2. 5-coder technical report," *arXiv preprint arXiv:2409.12186*, 2024.

[155] Zhe Chen, Weiyun Wang, Yue Cao, *et al.*, "Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling," *arXiv preprint arXiv:2412.05271*, 2024.

[156] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 213–229.

[157] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun, "Anchor detr: Query design for transformer-based detector," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, 2022, pp. 2567–2575.

[158] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1492–1500.

[159] Wolfgang Postl, "Detection of linear oblique structures and skew scan in digitized documents," in *Proc. Int. Conf. on Pattern Recognition*, 1986, pp. 687–689.

[160] Devashish Prasad, Ayan Gadpal, Kshitij Kapadni, Manish Visave, and Kavita Sultanpure, "Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 572–573.

[161] Khurram Azeem Hashmi, Didier Stricker, Marcus Liwicki, Muhammad Noman Afzal, and Muhammad Zeshan Afzal, "Guided table structure recognition through anchor optimization," *IEEE Access*, vol. 9, pp. 113 521–113 534, 2021.

[162] Zhenrong Zhang, Jianshu Zhang, Jun Du, and Fengren Wang, "Split, embed and merge: An accurate table structure recognizer," *Pattern Recognition*, vol. 126, p. 108 565, 2022.

[163] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes, "Image-based table recognition: Data, model, and evaluation," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 564–580.

[164] Nam Tuan Ly and Atsuhiro Takasu, *An end-to-end multi-task learning model for image-based table recognition*, arXiv:2303.08648, Preprint, 2023.

[165] Chunkai Ding, "Financial statement recognition based on support vector machine method," in *Cyber Security Intelligence and Analytics: 5th International Conference on Cyber Security Intelligence and Analytics (CSIA 2023), Volume 1*, ser. Lecture Notes on Data Engineering and Communications Technologies, Springer, 2023, pp. 160–169.

[166] Rongxu Liu, "Applications of new technologies to recognition of financial statement fraud," in *2020 International Conference on Computer Communication and Network Security (CCNS)*, IEEE, 2020, pp. 22–25.

[167] "Financial accounting, reporting and analysis," in *Financial Accounting, Reporting and Analysis*, Oxford University Press, 2017. DOI: 10.1093/hebz/9780198745310.003.0025.

[168] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee, "Visual instruction tuning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, Curran Associates, Inc., 2023, pp. 34 892–34 916.

[169] Geewook Kim, Teakgyu Hong, Moonbin Yim, *et al.*, "Ocr-free document understanding transformer," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, Springer, 2022, pp. 498–517.

[170] Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen, "OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2022, pp. 932–942. DOI: 10.18653/v1/2022.naacl-main.68.

[171] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao, "Unified vision-language pre-training for image captioning and vqa," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 13 041–13 049.

[172] Nils Reimers and Iryna Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410.

[173] Yoonsik Kim, Moonbin Yim, and Ka Yeon Song, *Tablevqa-bench: A visual question answering benchmark on multiple table domains*, arXiv:2404.19205, Preprint, 2024.

[174] Ali Furkan Biten, Ruben Tito, Andres Mafla, *et al.*, "Icdar 2019 competition on scene text visual question answering," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 1563–1570.

[175] Rubèn Tito, Minesh Mathew, CV Jawahar, Ernest Valveny, and Dimosthenis Karatzas, "Icdar 2021 competition on document visual question answering," in *Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part IV 16*, Springer, 2021, pp. 635–649.

[176] Awni Hannun, Jagrit Digani, Angelos Katharopoulos, and Ronan Collobert, *MLX: Efficient and flexible machine learning on apple silicon*, version 0.0, 2023. [Online]. Available: https://github.com/ml-explore.

[177] Prince Canuma, *MLX-VLM: Mlx-vlm is a package for inference and fine-tuning of vision language models (vlms) on mac using mlx.* Version 0.1.14, 2024. [Online]. Available: `https://github.com/Blaizzy/mlx-vlm`.

[178] Sarthak Jain and Byron C Wallace, "Attention is not explanation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 3543–3556.

[179] Sarah Wiegreffe and Yuval Pinter, "Attention is not not explanation," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 11–20.

[180] Samira Abnar and Willem Zuidema, "Quantifying attention flow in transformers," in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 4190–4197.

[181] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[182] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly, "Pointer networks," *Advances in neural information processing systems*, vol. 28, 2015.