A Holistic Framework for Multimodal Ecosystem of Pictionary

Thesis submitted in partial fulfillment
of the requirements for the degree of

*(Master of Science in **Computer Science and Engineering** by Research)*

by

Nikhil Bansal
20161065
nikhil.bansal@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
September, 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "A Holistic Framework for Multimodal Ecosystem of Pictionary" by Nikhil Bansal, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Ravi Kiran Sarvadevabhatla

To, my family

Acknowledgments

I would like to express my gratitude to all of the people who supported and guided me. First and foremost, my deepest and sincerest gratitude to my mentor, teacher and guide **Dr. Ravi Kiran Sarvadevabhatla**, Assistant Professor, International Institute of Information Technology (IIIT-H), Hyderabad, for his immense guidance and encouragement throughout the course of this research. And above all his unwavering attitude to aid his students, providing all the necessary tools, reference works, and research papers have been invaluable in shaping my thesis.

I would also like to express my sincere thanks to Dr. Naresh Manwani, who along with Dr. Ravi Kiran helped me gain thorough understanding of fundamentals through several brainstorming and enlightening discussions pertaining to our research problem.

I am also grateful to my colleagues Kiruthika Kannan, Kartik Gupta for their helpful discussions and ideas that have contributed to my research work. I would also like to express my appreciation to my friends Sparsh Garg, Piyush Kumar, Shubham Pareek and Mohit Sharma, for constantly motivating me and helping me balance my work and college life. I have shared some of my most memorable moments of my college life with them and I am really fortunate to have them as my friends.

Finally, I am also highly indebted to my family for their unwavering support, love, and encouragement during the time of research. Their support has been crucial in helping me to complete this work.

Abstract

In AI, the ability of intelligent agent to model human player in games such as Backgammon, Chess and Go has been an important metric in benchmarking progress. Fundamentally, the games mentioned above can be characterized as competitive and zero-sum. In contrast, games such as Pictionary and Dumb Charades falls into the category of 'social' games. Unlike competitive games, the emphasis is on cooperative and co-adaptive game-play in a relaxed setting. Such social games can form the basis for the next wave of game-driven progress in AI. Pictionary™ is a wonderful example of cooperative game play to achieve a shared goal in communication-restricted settings. This popular sketch-based guessing game, which we employ as a use case, provides an opportunity to analyze shared goal cooperative game play in restricted communication settings.

To enable the study of Pictionary and to understand various aspects associated with the game play, we designed a software ecosystem for web-based online game of Pictionary dubbed PICTGUESS. To overcome several technological and logistic barriers, which the actual game presents, we implemented a simplified setting for PICTGUESS wherein a game consists of a time-limited episode involving two players - a Drawer and a Guesser. The Drawer is tasked with conveying a given target phrase to a counterpart Guesser by sketching on a whiteboard within that time limit.

However, occasionally some players in Pictionary draw atypical sketch content. While such content is occasionally relevant in the game context, it sometimes represents a rule violation and impairs the game experience. To address such situations in a timely and scalable manner, we introduce DRAWMON, a novel distributed framework for automatic detection of atypical sketch content in concurrently occurring Pictionary game sessions. We build specialized online interfaces to annotate atypical sketch content, resulting in ATYPICT, the first ever atypical sketch content dataset. We use ATYPICT to train CANVASNET, a deep neural atypical content detection network. We utilize CANVASNET as a core component of DRAWMON. Our analysis of post deployment game session data indicates DRAWMON's effectiveness for scalable monitoring and atypical sketch content detection. Beyond Pictionary, our contributions can also serve as a design guide for customized atypical content response systems involving shared and interactive whiteboards.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the history of Artificial Intelligence, computer-based modelling of human player games such as Backgammon, Chess and Go has been an important research area. The accomplishments of well-known game engines (e.g. DeepBlue, AlphaGo) and their ability to mimic human-like game moves has been a well-accepted proxy for gauging progress in AI. Fundamentally, the games mentioned above can be characterized as competitive and zero-sum(one player loses while the other wins). In contrast, there are other games such as Pictionary and Dumb Charades which fall into the category of 'social' games. Unlike competitive games, the emphasis is on cooperative and co-adaptive game-play in a relaxed setting. Such social games can form the basis for the next wave of game-driven progress in AI. However, such games need a rethink in terms of agent and game play modelling. In this work, we explore the design space for such games using pictionary as a use-case. The game of pictionary uses a shuffled deck of cards with guess-words printed on them. The participants first group themselves into teams and each team takes turns. For a given turn, a team's member selects a card. He/She then attempts to draw a sketch corresponding to the word printed on the card in such a way that the teammates can guess the word correctly. The rules of the game forbid any verbal communication between the drawer and team-mates. Thus, the drawer conveys the intended guess-word primarily via the sketching process.

It is important to contrast pictionary with the typical two-player, zero-sum games for which AI agents exist. Pictionary is complicated by additional factors such as multi-modal game play (guesser uses speech/lexical modality while drawer uses visual modality), asynchronous turn-taking and only high-level notions of what constitutes a 'win' (e.g. target word is 'airplane', but guesser's proposal of 'warplane' may be considered acceptable). Since our objective is to analyze and explore design space of pictionary from computer-based modelling perspective, it is important to collect a large repository of actual game sessions. However, data collection for the actual game setting has several technological and logistic barriers. Therefore, we propose a simplified yet realistic setting wherein a game consists of a time-limited episode involving two players - a Drawer and a Guesser. The Drawer is tasked with conveying a given target phrase to a counterpart Guesser by sketching on a whiteboard [14]. The larger the number of target phrases correctly identified and the earlier the phrases are identified from the drawn sketch, greater the number of points accrued for the participating players.

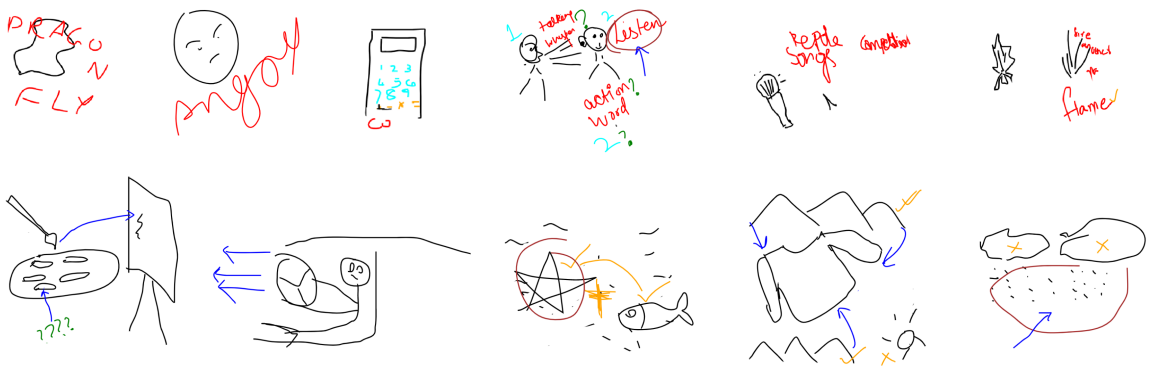

Figure 1.1: Some examples of atypical sketch content in Pictionary game sessions are shown as canvas screenshots.

The rules of Pictionary forbid the Drawer from writing text on the whiteboard. This is usually not an issue when players are physically co-located. In the anonymized, web-based version of the game, however, the Drawer may cheat by writing text related to the target word on the digitally shared whiteboard, thus violating the rules. Apart from malicious game play, atypical sketch content(such as text) can also exist in non-malicious, benign scenarios. For instance, the Drawer may choose to draw arrows and other such icons to attract the Guesser's attention and provide indirect hints regarding the target word (see Fig. 1.1).

1.1 Thesis Contributions

In this thesis, we address the requirement for a framework which can respond to a variety of atypical whiteboard sketch content in a reliable, comprehensive and timely manner. To this end, we make the following contributions:

- We introduce PICTGUESS - a web-based online game based on a realistic two-player setting of the game of Pictionary.
- We introduce ATYPICT - the first ever dataset of atypical whiteboard content, collected using PICTGUESS.
- We introduce CANVASDASH - a custom-designed, browser-based annotation and visualization tool to annotate atypical whiteboard content.
- We introduce CANVASNET - inspired by the success of deep networks which attempt to detect text in photos [29, 44, 27], we adopt a similar efficient approach for our CANVASNET deep network to detect atypical sketch instances on a drawing canvas.

- We introduce DRAWMON - a distributed system for sketch content-based alert generation. We analyze sessions with DRAWMON deployed for Pictionary setting and demonstrate its effectiveness.

1.2 Challenges

1.2.1 Data collection

Existing sketch datasets (e.g. TU-Berlin [11], Sketchy [40], QuickDraw [21]) have been created primarily in the context of sketch *object* recognition problem – assign a categorical label to a hand-drawn sketch. The category labels correspond to objects (nouns). Therefore, these datasets lack abstract sketches which tend to be drawn when words from other parts of speech (verbs, adjectives) are provided as targets. Existing datasets are also unnatural because they do not include canvas actions such as erase strokes or location emphasis. Also, no intermediate guess words are associated with sketched content. For a similar reason, these datasets do not contain atypical activities unlike the dataset we introduce. Sarvadevabhatla et al. [42] explore neural network based generation of human-like guesses, but for pre-drawn object sketches. However, they do not accommodate interactivity and non-sketch drawing canvas activities (e.g. erase, pointing emphasis). The Kondate dataset [32] contains on-line handwritten patterns of text, figures, tables, maps, diagrams etc. The OHFCD dataset [2] pertains to online handwritten flowcharts. Although challenging in their own way, these datasets are considerably more structured than our setting. Additionally, they share the sketch datasets' shortcoming of being too cleanly curated because actions such as erase are absent. To address these challenges, we collected our data using PICTGUESS - our custom designed browser-based game portal to play simplified setting of Pictionary, which gives participants ability to erase, highlight or make multiple guesses during game sessions. As a unique aspect, our combination of such game settings and a fixed time limit per game session unleashes greater diversity and creativity, causing sketches in our dataset to be more spontaneous and less homogeneous compared to existing datasets.

1.2.2 Manual Annotation of Sketch Data

As mentioned earlier, existing sketch datasets assign a categorical label to a hand-drawn sketch which makes them unsuitable for the purpose of detecting the exact position and categories(3.1) of atypical content. To be able to detect aforementioned categories of atypical content in a sketch, manual annotation of each stroke of a sketch as belonging to one of the category is a necessity.

1.2.3 Scalable and time-efficient Framework for sketch content-based alert generation

Consider a scenario with multiple online Pictionary game sessions in progress. We require a framework for automatic and concurrent monitoring of these game sessions for any atypical activities and

issue instant alerts to the multiple end users for rule violation(e.g. such as writing text on canvas). To meet these requirements, a framework needs to be reliable, scalable and time-efficient.

1.3 Thesis Layout

The compilation of thesis is as follows.

Chapter 2 discuss PICTGUESS, a software ecosystem for web-based online game based on a realistic two-player setting of the game of Pictionary.

Chapter 3 Provides insights into ATYPICT - our custom dataset of atypical whiteboard content. All major components of ATYPICT are explained in detail along with the proposal of tools for data annotation.

Chapter 4 discuss CANVASNET, a novel deep neural network to detect atypical sketch content on a drawing canvas. A detailed explanation of network architecture and training methodology is provided with quantitative and qualitative results.

Chapter 5 discuss DRAWMON, a framework for automatic, concurrent monitoring of pictionary game sessions for any atypical activities. It also discusses the brief user-study to quantify the efficacy of DRAWMON. Detailed description of the user-study setup and observations are also provided.

Chapter 6 concludes this thesis by consolidating all the contributions of this thesis. We also discuss some potential extensions of our system for the researchers working in this field to pursue.

Chapter 2

PICTGUESS: Software Ecosystem for Web-based Online Game of Pictionary

The social game Pictionary, is a wonderful example of cooperative gameplay in a communication restricted settings [15, 51, 38] and it also presents scenarios involving atypical sketched content. Unlike other zero-sum games, pictionary is complicated by additional factors such as multi-modal game play, asynchronous turn-taking and only high-level notions of what constitutes a 'win'. Due to above mentioned reasons we employ Pictionary as a use case for exploring design space of such social games. For doing that, it is important to collect a large repository of actual game sessions. However, data collection for the actual game setting has several technological and logistic barriers. Therefore, we propose a simplified setting wherein a game consists of a time-limited episode involving two players - a Drawer and a Guesser. The Drawer is tasked with conveying a given target phrase to a counterpart Guesser by sketching on a whiteboard [14]. The larger the number of target phrases correctly identified and the earlier the phrases are identified from the drawn sketch, greater the number of points accrued for the participating players. For playing the above described simplified setting of Pictionary to collect data, we designed a custom browser-based game portal dubbed PICTGUESS.

2.1 PICTGUESS

2.1.1 System Design

After carefully assessing offline Pictionary game and end users needs, we gathered design requirements for our 2 player simplified setting of Pictionary (See Figure 2.1). To realise the requirements shown in Figure 2.1, we designed PICTGUESS to be compatible with mouse and touch inputs. We obtain consent and provide game instructions when a player accesses the system for the first time. Our system is scalable and can handle up to 50 multiple concurrent sessions. Players are assigned random names and paired randomly as Drawers and Guessers.

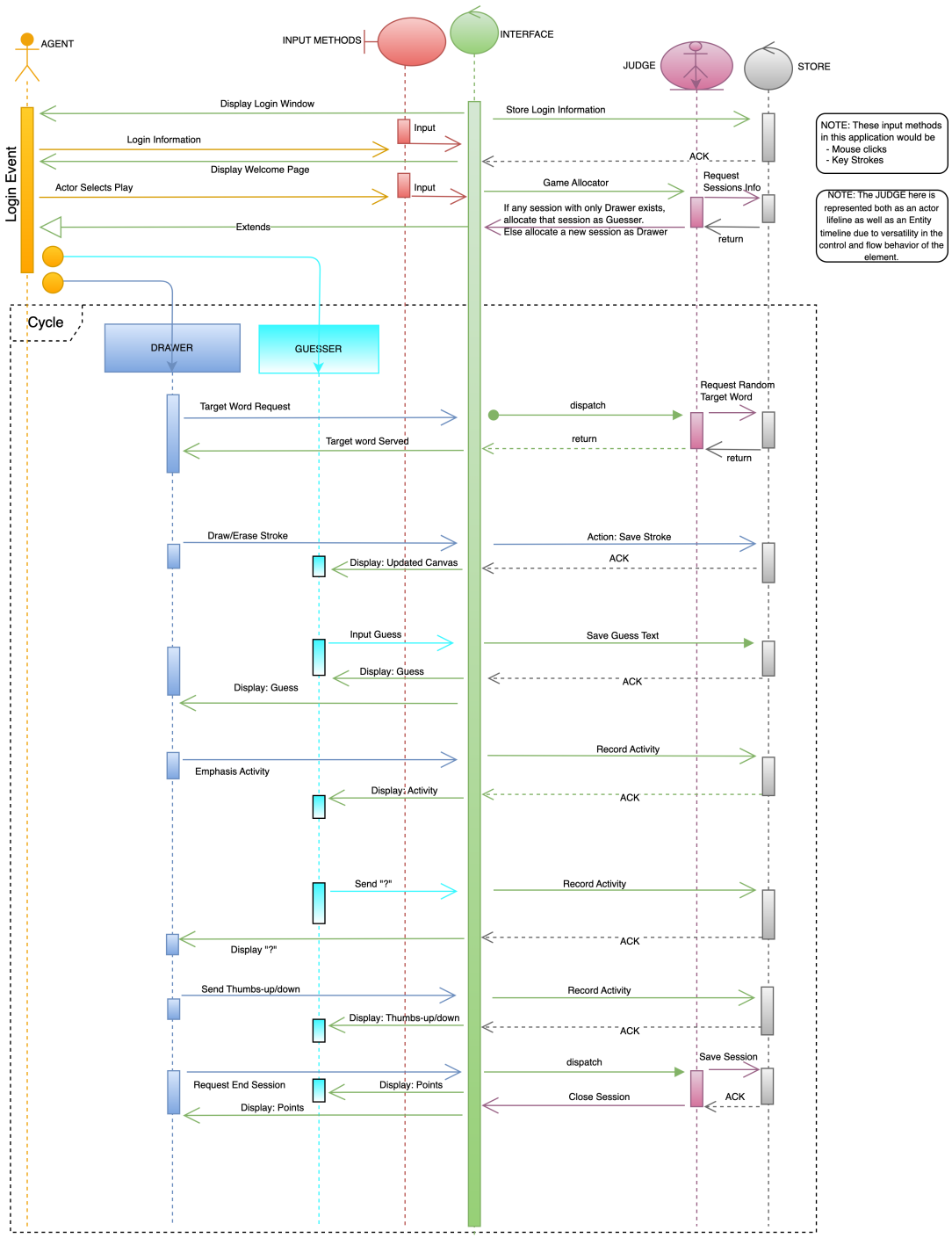

Figure 2.1: System model for online Pictionary game.

For the Guesser, a text box is provided for entering guess phrases. For the Drawer, the interface provides a canvas with tools to draw, erase and highlight locations (via a time-decaying spatial animation 'ping') for emphasis (see Fig. 2.2). The targets provided to the Drawers are sampled from a dictionary

of 200 guess phrases. We re-emphasize that the target phrases can be nouns (e.g. airplane, bee, chair), verbs (e.g. catch, call, hang) or adjectives (e.g. happy, lazy, scary). To ensure uniform coverage across the dictionary, the probability of a guess phrase being selected for a session is inversely proportional to the number of times it has been selected for elapsed sessions. The game has a time limit of 120 seconds. The game ends when the Guesser enters a word deemed 'correct' by the Drawer or when the time limit is reached. In addition, 👍 and 👎 buttons enable Drawer to provide 'hot/cold' feedback on guesses. For instance, if the Guesser's guess is close to the target word (e.g. 'pen' for the target word 'pencil'), the Drawer might provide 👍 as feedback. A question (⍰) button is provided to the Guesser for conveying that the canvas contents are not informative and confusing. The canvas strokes are timestamped and stored in Scalable Vector Graphic (SVG) format for efficiency. In addition to canvas strokes (drawing and erasure related), secondary feedback activities mentioned previously (👎 ,👍 , ⍰, highlight) are also recorded with timestamps as part of the game session. See Figure 2.2 for screenshots from our game portal.

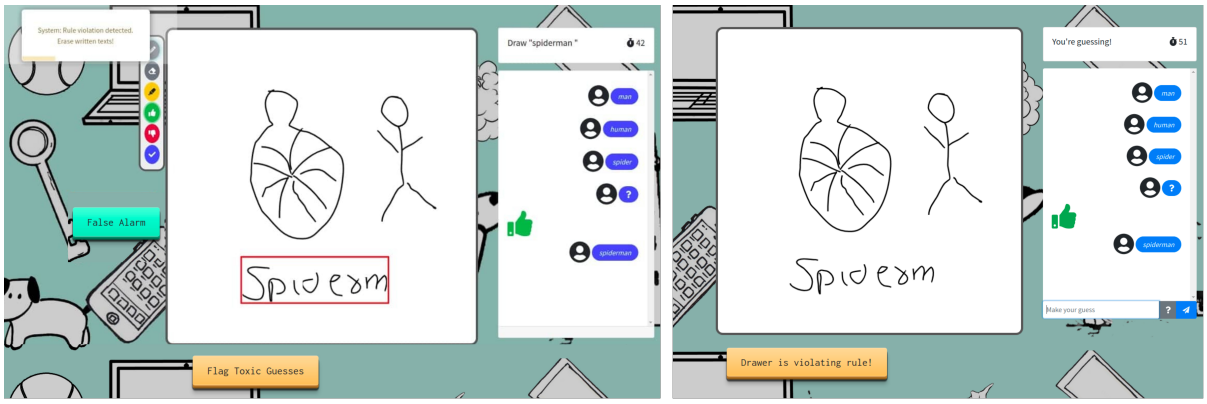

Figure 2.2: Screenshots of our game portal showing Drawer (left) and Guesser (right) activity during a Pictionary game.

2.1.2 System Architecture

Figure 2.3 shows the system architecture of our browser-based game portal dubbed PICTGUESS. Central Relay Server is responsible for pairing Drawer and Guesser from the pool of available players and receiving and relaying information from browser client to Session Manager and Data Relay Server. Each game session is managed by a central Session Manager which assigns a unique session id. For a given session, whenever a sketch stroke is drawn, the accumulated canvas content (i.e. strokes rendered so far) is timestamped and stored in SVG format in database via Data Relay Server. Data Relay Server is also responsible for timestamping and storing guesses for a session in database.

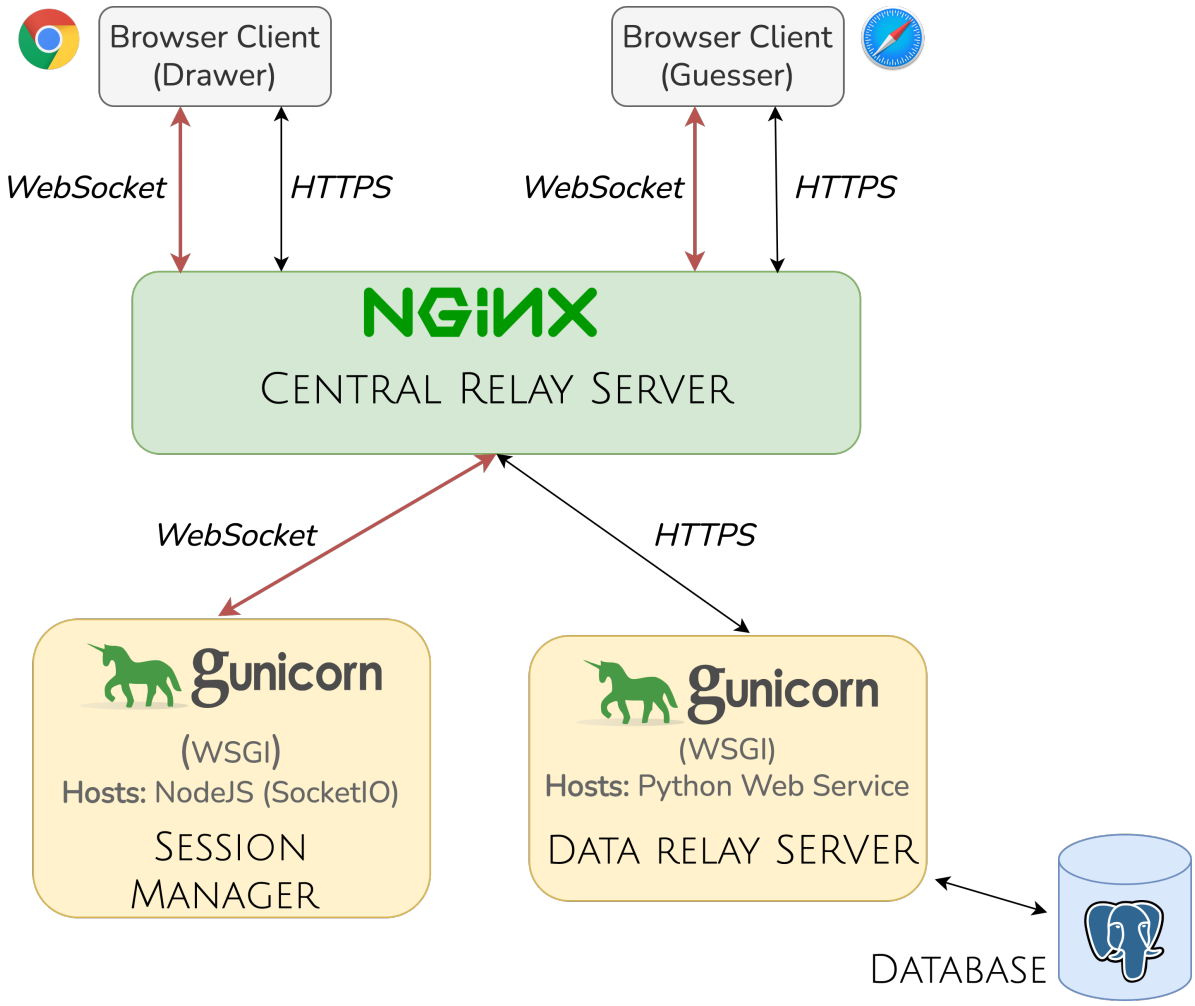

Figure 2.3: System architecture for Pictionary game setup.

2.1.3 Data Collection

Via our portal, we successfully gathered 3220 timestamped episodes of diverse, realistic game play involving a total of 479 participants in a large age range (14 years to 60 years) and educational demographics (middle and high school students, graduate and undergraduate university students and working professionals). See Table 2.1 for additional statistics of the collected data.

2.2 Observations

Our combination of game settings in PICTGUESS such as time limit, highlight, erase tool, category labels from all parts of speech (nouns, verbs, adjectives), capability to record timestamped guesses and strokes (for associating guesses with sketched content) tackles all the limitations (no abstract sketches as no target words from other parts of speech such as verbs or adjectives, no erase strokes or location

Table 2.1: Statistics of dataset

Number of sessions collected	3220	
Number of users	479	
Number of target words	200	
Noun		138
Verb		51
Adjectives		11

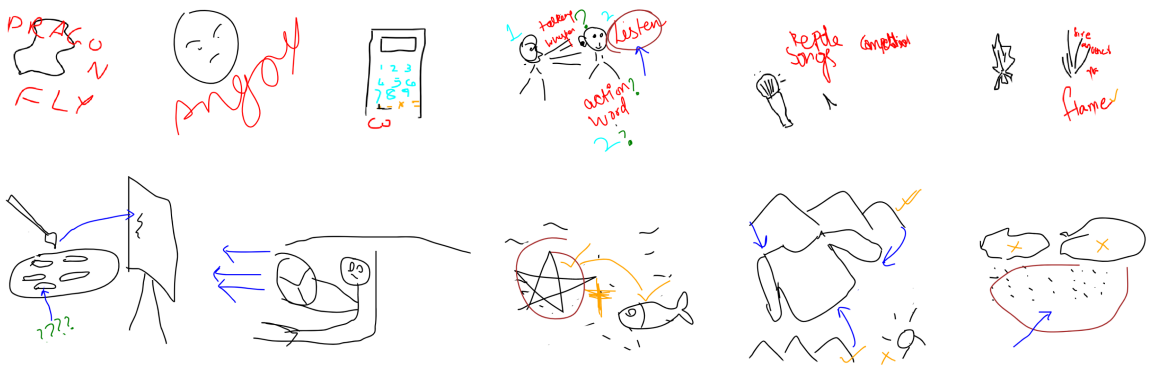

Figure 2.4: Some examples of atypical sketch content in Pictionary game sessions are shown as canvas screenshots. The content instances span text highlighted in red, numbers (cyan), question marks (green), arrows (blue), circles (Maroon) and other icons (e.g. tick marks, addition symbol) highlighted in orange.

emphasis, no intermediate guess words) of existing datasets (e.g. TU-Berlin [11], Sketchy [40], Quick-Draw [21]) and also unleashes greater diversity and creativity, causing sketches in our dataset to be more spontaneous and less homogeneous compared to existing datasets.

On carefully analysing our sketch dataset, we detected significant instances of malicious gameplay where Drawer cheated by writing text related to the target word on the digitally shared whiteboard, thus violating the rules. Apart from malicious game play, we also detected sketch contents that are only used as hints for Guesser to guess the target word. For instance, the Drawer may choose to draw arrows and other such icons to attract the Guesser's attention and provide indirect hints regarding the target word (see Fig. 2.4). In next chapter, we shall discuss our data annotation procedure to annotate such sketch contents which when applied on the above collected data (Table 2.1) resulted in ATYPICT - our custom dataset of atypical whiteboard content.

Chapter 3

ATYPICT: Dataset of Atypical Whiteboard Content

Shared digital whiteboards are becoming increasingly popular in educational and workplace settings as a natural mechanism for collaboration and communication [20, 13, 8, 34, 1, 24]. The sharing aspect offers tremendous scope for interaction and a richer session experience. Unfortunately, shared whiteboards sometimes present situations where malicious participants draw controversial content [49]. Such activities tend to impair the collective experience of participants. Therefore, it is important to have scalable system for efficiently identifying and responding to such activities.

Pictionary™, which we employ as a use case, also presents such scenarios. The rules of Pictionary forbid the Drawer from writing text on the whiteboard. This is usually not an issue when players are physically co-located. In the anonymized, web-based version of the game, however, the Drawer may cheat by writing text related to the target word on the digitally shared whiteboard, thus violating the rules. Apart from malicious game play, atypical sketch content (such as text) can also exist in non-malicious, benign scenarios. For instance, the Drawer may choose to draw arrows and other such icons to attract the Guesser's attention and provide indirect hints regarding the target word. To create a system to detect such atypical content in Pictionary, we need to annotate our sketch dataset. Next, we will first classify atypical activities and then we will discuss our annotation procedure.

3.1 Atypical Data

Atypical content can be thought of as a subsequence of sketch curves relative to the larger sequence of curves that comprise the game session. By carefully analyzing our sketch dataset, we categorized atypical content usually encountered in Pictionary sessions as follows:

- *Text*: Drawer directly writes the target word or hints related to the target word on the canvas.
- *Numerical*: Drawer writes numbers on canvas.
- *Circles*: Drawers often circle a portion of the canvas to emphasize relevant or important content.

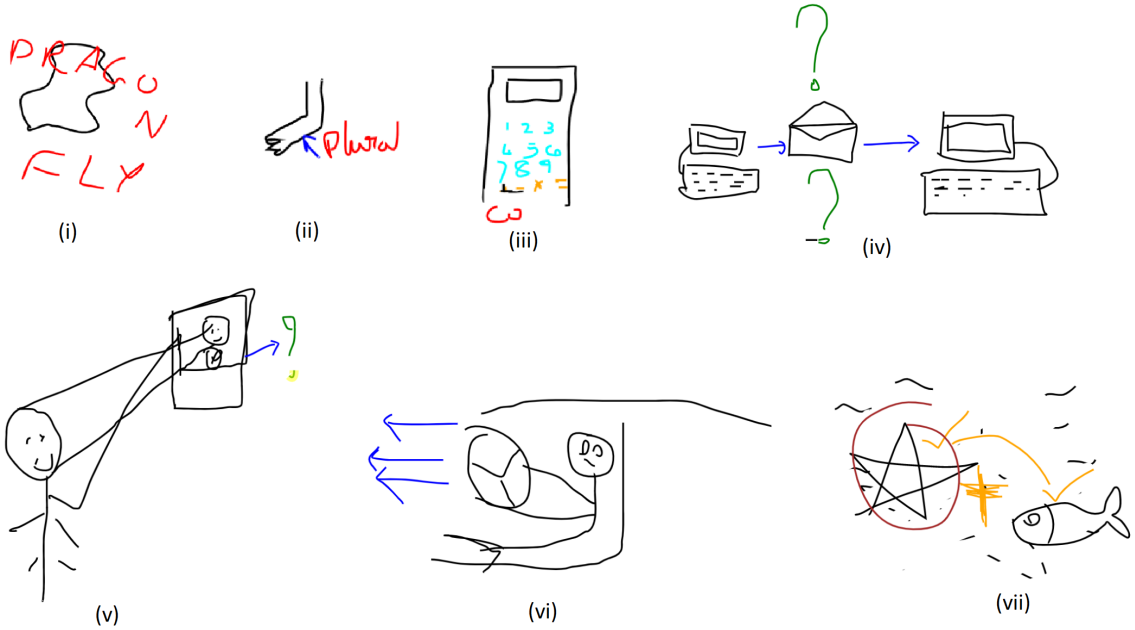

Figure 3.1: Examples of atypical content text, numbers, question marks, arrows, circles and other icons (e.g. tick marks, addition symbol).

- *Iconic*: Other items used for emphasizing content and abstract compositional structures include drawing a question mark, arrow and other miscellaneous structures (e.g. double-headed arrow, tick marks, addition symbol, cross) and striking out the sketch (which usually implies negation of the sketched item).

Examples can be viewed in Fig. 3.1. It is important to remember that we consider only *Text* writing as a rule violation in Pictionary. Other categories mentioned above are atypical but their presence is not considered a violation of game rules.

3.2 Data Annotation

Having identified the different types of atypical data, the question then arises, "how do we detect them?". To enable the creation of a system which can detect atypical activities, we need to annotate the collected data. To facilitate this annotation, we built our custom-designed, browser-based annotation and visualization tool dubbed CANVASDASH (see Fig. 3.2). The annotation interface displays strokes of a session in the temporal order in which they were recorded. The annotator can navigate through strokes using arrow keys. To assign labels, the annotator can choose from among the categories present in the menu bar (denoted 'Category Choices') at the bottom of the interface.

CANVASDASH also enables annotators to split a single sketch stroke into multiple sub-strokes. For example, a word written in running hand can be split into individual letters and annotated (See Figure

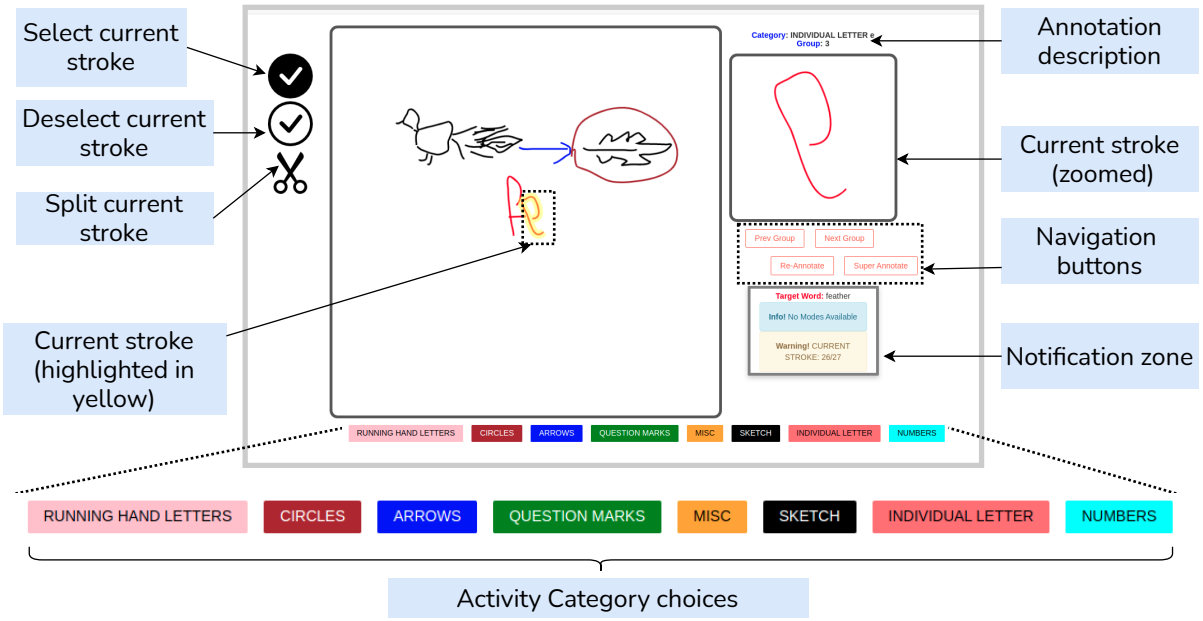

Figure 3.2: An illustration of the annotation of a session using the CANVASDASH interface. 'RUNNING HAND LETTERS', 'CIRCLES', 'QUESTION MARKS', 'MISC'(Miscellaneous), 'SKETCH'(regular sketch strokes), 'INDIVIDUAL LETTERS' and 'NUMBERS' are the canvas activity categories. Each of these categories are further classified under 4 broad atypical categories (see table-3.1, section-3.1). 'Select current stroke', 'Deselect current stroke' and 'Split current stroke' are used for grouping and splitting SVG curves. The current stroke/group is highlighted in the main canvas and zoomed in the side canvas to assist the annotators.

3.4). By clicking on the split-stroke button, a point appears at the beginning of the stroke. The point can be moved on the stroke using arrow keys. The current position of the point can be marked as a split point by pressing select button. After all the split points (being highlighted by the tool) are marked on the stroke, pressing the split-stroke button will split the stroke accordingly.
To enable instance-level annotation for a category consisting of multiple strokes (e.g. a word written as a combination of running hand and isolated letters), the interface provides an option for selecting a subsequence of strokes. To assist annotators with such multi-stroke annotation, the tool highlights each stroke that belongs to the selected subsequence (see Fig. 3.3).

For ease of verification and correction, a replay mechanism is also included which color-codes and shades strokes based on annotation category label (see Fig. 3.2). Atypical activities tend to be infrequent. Therefore, all strokes are initially labelled as valid sketch strokes. The session stroke sequence is consequently navigated to identify and label atypical activities as described above.

Figure 3.3: Illustration of the multi-stroke annotation feature. The annotator has grouped strokes 'M', 'icrop', 'h' and 'one' belonging to the word 'Microphone'. To assist the annotator, the tool has highlighted each selected stroke.

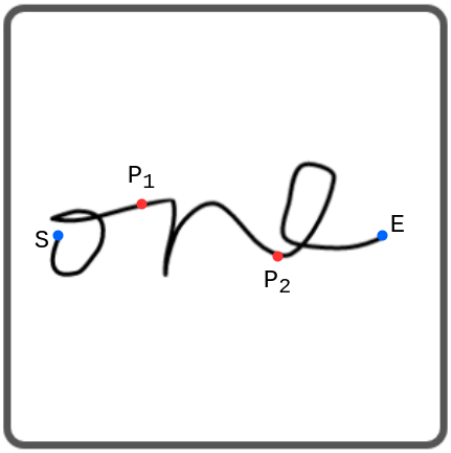

Figure 3.4: Illustration of the split-stroke feature. The stroke SE is shown with starting point S, and ending point E. The two marked split points P_1, P_2 splits the stroke into three parts SP_1, P_1P_2 and P_2E

3.3 Summary

Using the described annotation procedure, we obtain our atypical Pictionary sketch dataset ATYPICT. The occurrence statistics of atypical sketch categories across game sessions can be viewed in Table 3.1. Representative visual examples can be viewed in Fig. 3.1. Accurately localizing such activities can aid statistical learning approaches which associate sketch-based representations with corresponding target words [42]. In next chapter, we will describe CANVASNET, a novel deep neural network for detecting such atypical whiteboard sketch content.

Sketch Content Type class	Number of occurrences	Number of sessions containing	Number of target phrases containing
Text	**2419**	**478**	**180**
Individual letter	2244	460	178
Running hand	175	103	81
Numbers	**331**	**73**	**28**
Circles	**110**	**90**	**67**
Iconic	**750**	**377**	**147**
Arrow	497	292	129
Question mark	158	116	78
Miscellaneous	95	54	37

Table 3.1: Statistics of atypical sketch content categories in game sessions.

Chapter 4

CANVASNET: A Deep Neural Network for Atypical Activity Detection

4.1 Introduction

Accurately localizing atypical activities can aid statistical learning approaches which associate sketch-based representations with corresponding target words [42]. Considering both malicious and benign scenarios in Pictionary, the broad requirement is for a framework which can detect and flag a variety of atypical whiteboard sketch content in a reliable, comprehensive and timely manner. Flagging is possible by physically monitoring game sessions. However, such manual intervention is impractical and not scalable to an online setting involving a large number of multiple concurrent game sessions. Providing user interface options for player-triggered flagging of rule violation is another possibility. But such mechanisms are not completely reliable since the Guesser benefits from the content written on the canvas and does not have real incentive to use the flagging mechanism.

Some approaches employ a diverse mix of techniques for detecting cheating in online games [19, 48]. Dinh et al. [10] use hand-crafted game features and unsupervised machine learning approaches for offline detection of anomalous behavior. In our work, we introduce automatic deep learning based detection and flagging of anomalous gameplay in Pictionary. However, our system is also designed to detect secondary non-anomalous canvas entities which can potentially aid statistical understanding of canvas contents. Detecting those canvas entities can be thought of as a stroke segmentation problem wherein each sketch stroke is labelled as either belonging to an atypical class or the default class (drawing). Stroke segmentation has been employed for labelling parts in object sketches either from stroke sequence information [56, 53, 23, 35] or within an image canvas [54, 26].

Recognizing atypical sketch content can also be posed as an object detection problem. In this case, the objective is to obtain 2-D spatial bounding boxes enclosing sketch strokes corresponding to the atypical content. We adopt this approach because it is faster and more amenable to near real-time operation compared to segmentation. Handwritten text is the most common atypical sketch content class in Pictionary. Hence, it is reasonable to consider approaches solely designed for text detection in domains such as outdoor scenes and documents [18, 9, 31, 30, 27, 60, 3]. Similarly, detection-based

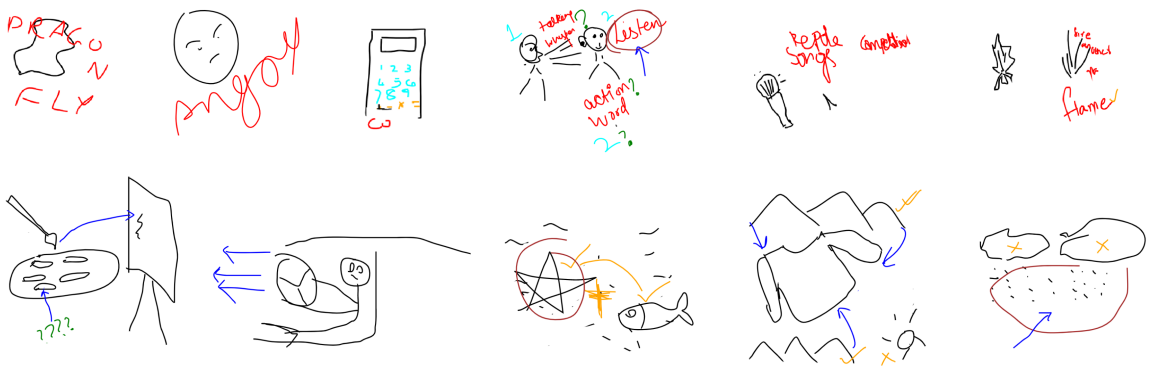

Figure 4.1: Some examples of atypical sketch content in Pictionary game sessions are shown as canvas screenshots. The content instances span text highlighted in red, numbers (cyan), question marks (green), arrows (blue), circles (Maroon) and other icons (e.g. tick marks, addition symbol) highlighted in orange.

approaches have been proposed for mixed graphic structures [22, 43, 12]. However, graphic elements in these scenarios are more structured compared to our Pictionary setting.

It is important to remember that we consider only *Text* writing as a rule violation in Pictionary. Other categories mentioned above are atypical but their presence is not considered a violation of game rules. Next, we will discuss our approach for detecting such atypical sketch content.

4.2 CanvasNet

An effective object detection approach for detecting atypical sketch instances needs to tackle the diversity in scale and appearance of various categories - a glance at Fig. 4.1 makes this amply clear. In addition, the approach needs to utilize spatial context and be robust to the presence of similar looking yet semantically distinct regular (sketch) canvas content. In our case, the drawing canvas containing accumulated sketch strokes is the image. Any atypical content instances (e.g. *Text*) present are considered spatially localized 2-D objects to be detected. For the object detection, we design a novel deep neural network which we dub CANVASNET. Before delving into details of CANVASNET, we first describe the data setup employed for its training and evaluation.

4.2.1 Data Preparation

As mentioned previously, the canvas elements for a given game session are represented as timestamped SVG curve elements. Each SVG element is either a drawing stroke or an erasure stroke. We group drawing strokes into subsequences which are separated by erase strokes. The curves are converted to a 2-D point sequence representation and adaptively downsampled into line-based strokes using Ramer–Douglas–Peucker [36] algorithm ($\epsilon = 2$). The strokes are rendered on a 512×512 2-D canvas

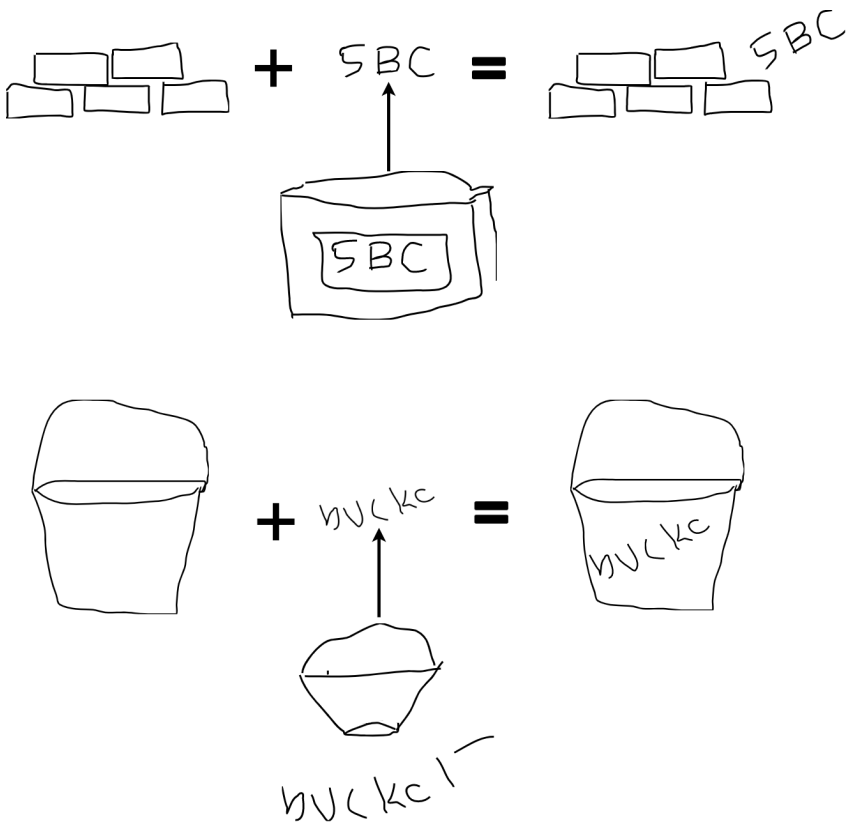

Figure 4.2: Examples of augmentation of atypical data. In each example, the image at bottom is the source image from which an instance of atypical activity is extracted and the left image is the one which is being augmented with that instance.

with a stroke thickness of 4 for the purpose of data annotation and representation. Note that a drawing stroke either belongs to one of the atypical classes (Sec. 3.1) or is a normal sketch stroke. The spatial extents of labelled stroke subsequences are used to automatically generate ground-truth data for training and evaluation. In our representation of each ground truth t, $G_t = (x_t, y_t, w_t, h_t)$ is the representation of the axis-aligned rectangle enclosing t, where (x_t, y_t) is the x, y coordinates of the center of the rectangle and w_t, h_t are the width and height of the rectangle.

Data Augmentation: The number of game sessions containing atypical instances are considerably smaller compared to the total number of game sessions. To increase the amount of data available for deep network training in a realistic manner, we first isolate atypical instance stroke subsequences. We sample from this set and add the resulting subsequences to other sessions which share the same target phrase, but do not contain any atypical content. The atypical content subsequences are also randomly rotated and localized carefully. This ensures they are spatially disjoint from strokes of the reference game sessions (which originally lack such atypical entities). Fig 4.2 shows examples of augmentation of atypical data.

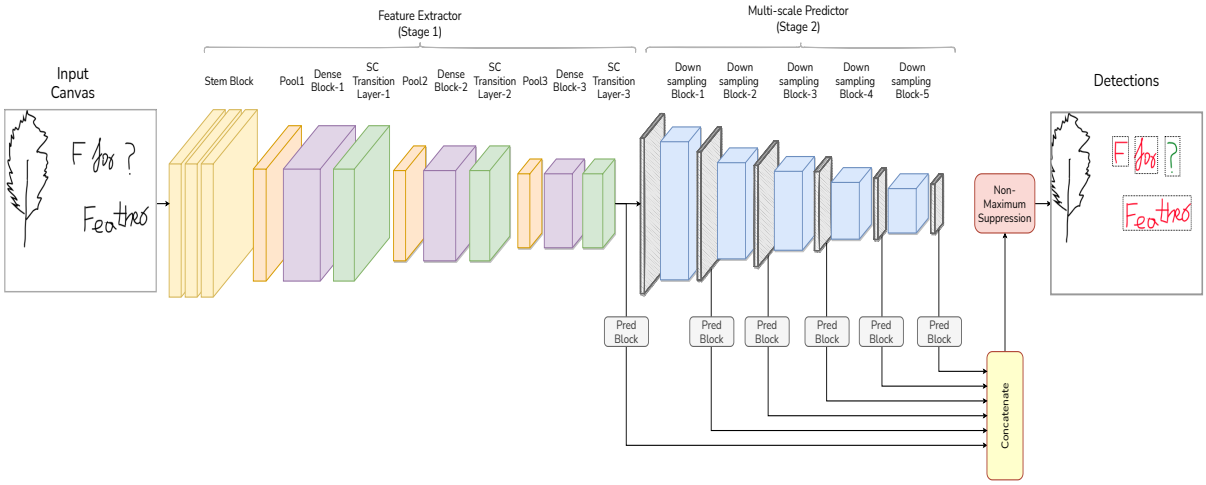

Figure 4.3: Architecture of CANVASNET deep neural network. Refer to Sec. 4.2.2 for details.

4.2.2 CANVASNET Deep Network

Inspired by the success of deep networks which attempt to detect text in photos [29, 44, 27], we adopt a similar efficient approach for our CANVASNET deep network to detect atypical sketch instances on a drawing canvas. CANVASNET consists of two stages.

Feature Extractor: The first stage consists of a stem block containing three 3×3 unit separable convolution layers [6]. Our choice of seperable convolution layers is motivated by the reduction in number of parameters and operations involved. The first layer in stem block uses a stride of 2 to downsample the input features. The stem block is then followed by a repeating three segment structure consisting of (i) a 2×2 max pooling layer (ii) a 6 layered dense block [17] with a growth rate of 48. Each of the 6 layers consists of a 1×1 separable convolution followed by a 3×3 separable convolution (iii) a 1×1 separable convolution layer denoted as SC Transition layer. This three segment structure is then repeated three times with similar parameters - see project page for additional details.

Multi-scale Predictor: Atypical content (e.g. handwritten text, arrows) can occupy varying amounts of drawing canvas area, to detect them we use multi-scale predictor for prediction on multiple scales of feature maps [27, 44]. For our setting, we use a customized multi-scale predictor for both handwritten text and non-text classes. The second stage sub-network is responsible for generating multi-scale feature maps and generating predictions over each of the feature maps. The output of third segment structure of the Feature Extractor is considered the first scale of the multi-scale feature map. The other feature map scales are obtained as outputs of successive downsampling blocks applied to the Feature Extractor's output (Fig. 4.3). The feature maps are passed individually through a prediction block. The multi-scale prediction features are concatenated and non-maximal suppression is applied to generate the final bounding box predictions.

Table 4.1: Detailed CANVASNET Architecture. Each dense block has growth rate $k = 48$ and $width = 4$. dep-conv and sep-conv denotes depthwise convolution and separable convolution.

Layers		Description
Stem	Separable convolution	3×3 sep-conv, stride 2, 96 filters
	Separable convolution	3×3 sep-conv, stride 1, 96 filters
	Separable convolution	3×3 sep-conv, stride 1, 96 filters
Dense Segment (1)	Pooling	2×2 max pool, stride 2
	Dense Block-1	$\begin{bmatrix} 1 \times 1 \text{ sep-conv} \\ 3 \times 3 \text{ sep-conv} \end{bmatrix} \times 6$, stride 1, $k = 48$, $width = 4$
	SC Transition Layer-1	1×1 sep-conv, stride 1, 224 filters
Dense Segment (2)	Pooling	2×2 max pool, stride 2
	Dense Block-2	$\begin{bmatrix} 1 \times 1 \text{ sep-conv} \\ 3 \times 3 \text{ sep-conv} \end{bmatrix} \times 6$, stride 1, $k = 48$, $width = 4$
	SC Transition Layer-2	1×1 sep-conv, stride 1, 224 filters
Dense Segment (3)	Pooling	2×2 max pool, stride 2
	Dense Block-2	$\begin{bmatrix} 1 \times 1 \text{ sep-conv} \\ 3 \times 3 \text{ sep-conv} \end{bmatrix} \times 6$, stride 1, $k = 48$, $width = 4$
	SC Transition Layer-2	1×1 sep-conv, stride 1, 224 filters
Downsampling Block (1)	Pooling	2×2 max pool, stride 2
	Depthwise Convolution	3×3 dep-conv, stride 2
	Separable Convolution	1×1 sep-conv, stride 1, 192 filters
Downsampling Block (2)	Pooling	2×2 max pool, stride 2
	Depthwise Convolution	3×3 dep-conv, stride 2
	Separable Convolution	1×1 sep-conv, stride 1, 160 filters
Downsampling Block (3)	Pooling	2×2 max pool, stride 2
	Depthwise Convolution	3×3 dep-conv, stride 2
	Separable Convolution	1×1 sep-conv, stride 1, 128 filters
Downsampling Block (4)	Pooling	2×2 max pool, stride 2
	Depthwise Convolution	3×3 dep-conv, stride 2
	Separable Convolution	1×1 sep-conv, stride 1, 96 filters
Downsampling Block (5)	Pooling	2×2 max pool, stride 2
	Depthwise Convolution	3×3 dep-conv, stride 2
	Separable Convolution	1×1 sep-conv, stride 1, 64 filters

Prediction block: This consists of a 3×5 separable convolution layer, followed by a fully connected layer comprising the prediction. The rectangular filter dimensions (3×5) used in the block ensure that elongated objects can be detected reliably.

Anchor boxes with vertical offsets: Among the atypical object categories, words have larger aspect ratios and range of box orientation. Therefore, we set anchor aspect ratios to $1, 2, 3, 4, 5, 1/2, 1/3, 1/5$ with ± 0.25 as the vertical offset.

Table-4.1 shows detailed CANVASNET Architecture.

Optimization: We formulate the loss function for CANVASNET as a combination of a classification loss L_{cls} and a bounding-box localization loss L_{loc}:

$$L = \alpha \left(\frac{1}{N} \sum_{i=1}^{N} L_{cls}(P_i, G_i) \right) + \left(\frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{c} G_{ij} * L_{loc}(B_i, B_i^{gt}) \right) \tag{4.1}$$

where G is the ground truth label matrix ($G_{ij} = 1$ if i-th anchor box belongs to atypical category j, else $G_{ij} = 0$), P is the predicted confidence score matrix (P_{ij} indicates the confidence score that i-th anchor box belongs to category j), this means P_i is a row of confidence scores for i-th box to belong to various atypical categories, similarly G_i is a row where values for all atypical category is 0 except for one(to which i-th box belongs). In the above loss function formulation, B_i^{gt} and B_i denotes the ground truth and predicted offsets for the i-th anchor box. N is the total number of anchor boxes, M is the total number of ground truth anchor boxes not belonging to background class, c represents the number of atypical categories. We use focal loss[28] for L_{cls} which prioritizes a sparse set of hard examples and prevents large number of negatives from overwhelming the detector. For localisation task, we adopt Distance-IoU Loss [58]:

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2\left(\mathbf{b}, \mathbf{b}^{gt}\right)}{c^2}$$

where b and b^{gt} denote the central points of predicted and ground truth bounding box, $\rho^2\left(\mathbf{b}, \mathbf{b}^{gt}\right)$ gives the square of the Euclidean distance between them, and c is the length of the diagonal of the smallest enclosing box covering the two bounding boxes. In object detection tasks, [58] shows the significant performance gains with their proposed CIoU and DIoU loss over other regression losses. We adopted DIoU loss over CIoU as the texts in our detection task can be very small and DIoU performs much better for small objects as shown in [58].

4.3 Baselines

All along, our approach for detecting atypical activities treats the canvas as a 2-D image. In effect, the game session is considered to be a video-like frame sequence of 2-D canvas images. We also consider alternate approaches wherein the game session is processed as a sequence of curves. Each curve is labelled either as a regular sketch stroke or one associated with an atypical content category.

To our best knowledge, there are no existing model to detect and localize atypical events in a canvas. Hence we compare the performance of our data using models designed for tasks like text detection, object detection and sketch segmentation which are similar to our problem. Table 4.4 shows the detection time for each baseline models in milliseconds.

4.3.1 BiLSTM+CRF

We first try the most straight forward model, Fig 4.4 which is inspired by [7]. We first extract a set of unary and pairwise features (refer Table 4.2) for each stroke similar to [50]. The features are

Table 4.2: Features extracted from strokes

Category	Feature
Unary features	
	Length of stroke
	Height of stoke
	Width of stroke
	Length of minor axis
	Length of major axis
	Eccentricity
	Rectangularity
	Area of convex hull
	Density
	Curvature
	Closure
	Time duration of stroke
Pairwise features	
	Ratio of length
	Ratio of width
	Ratio of height
	Ratio of area
	Intersection over union of area
	Distance between centroid
	Distance between endpoints
	Time Lapse between stroke

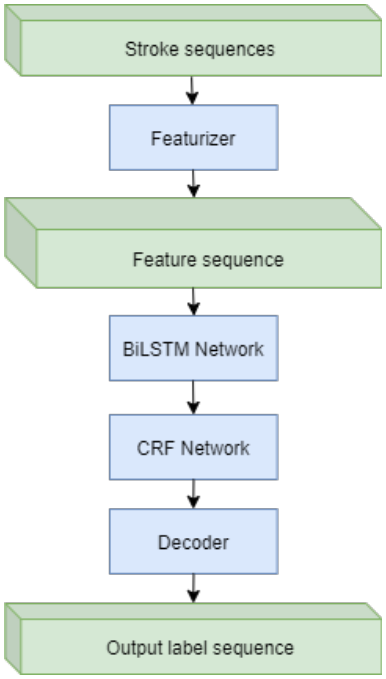

Figure 4.4: BiLSTM+CRF architecture

selected considering the characteristics of the atypical event in our dataset. The unary features are extracted from a single stroke while the pairwise features are calculated for the current and the previous strokes. Since our data is sequential we use a BiLSTM (BiDirectional Long Short Term Memory) layer to further encode the features. The output sequences of the BiLSTM are given to a Conditional Random Field (CRF) layer which is an undirected graphical model that predicts the conditional probability of the output classes. Finally the output classes are decoded using Viterbi algorithm.

4.3.2 SketchsegNet+

SketchsegNet+ [35] is a RNN based model for multi-class sketch semantic segmentation where each stroke is classified as one of the segmentation classes. Each point in the stroke are represented as a five dimensional vector $S = [\Delta x, \Delta y, p_1, p_2, p_3]$ where $[\Delta x, \Delta y]$ is the differential offset of a point coordinates from the previous points and $[p_1, p_2, p_3]$ are binary flags to mark an ongoing stroke, last point of stroke and end of the drawing respectively. A sequence to sequence Variational Auto Encoder (VAE) based model as shown in Fig 4.5 is used to generate an one-hot encoding of the output class for each point in the input. The input is first given to a BiLSTM encoder which predicts the mean and variance of a Gaussian latent vector variable. The sampled latent vector z is then given to a LSTM network along with the input S. Finally a set of fully connected layers followed by a softmax layer is used to decode the output labels from the LSTM output sequence. The model is trained to optimize the mean square loss between the predicted labels and the groundtruth. We calculate the evaluation metrics for each sequence of predicted class labels.

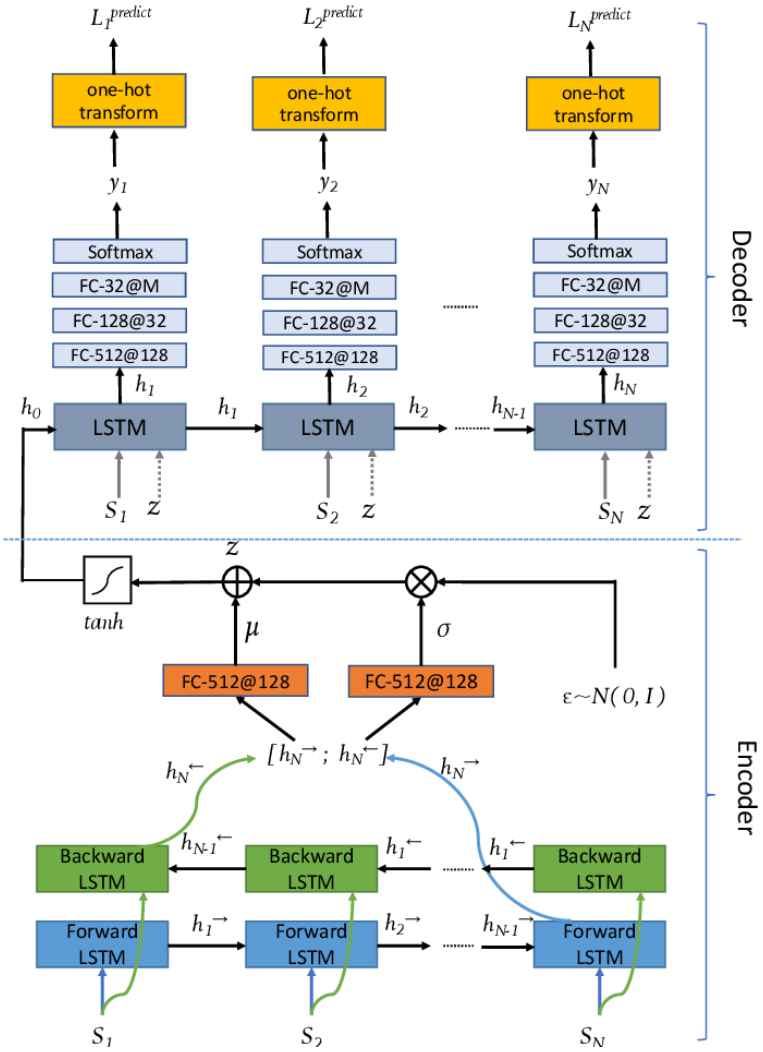

Figure 4.5: Sketchsegnet+ architecture

4.3.3 CRAFT

CRAFT [3] is a scene text detection method which produces region and affinity heatmaps for the input text, making them a perfect fit to detect text of arbitrary shape and orientation. Unlike text in natural settings, the text in sketch images are subjected to much more intrinsic variation and using CRAFT directly will not be a viable choice. So, the following architectural changes were made for effective learning.

The model adapts U-net [39] based architecture in the decoding block, similar to CRAFT. The ground truth labels are generated using Gaussian heatmaps at the center of every character in the input image.

1. The input images with respect to the game are relatively sparser than the generic scene text images. So the VGG-16 backbone of CRAFT is replaced with a simpler Sketch-a-Net [57] backbone. Additionally we made changes to the Sketch-a-Net backbone as well

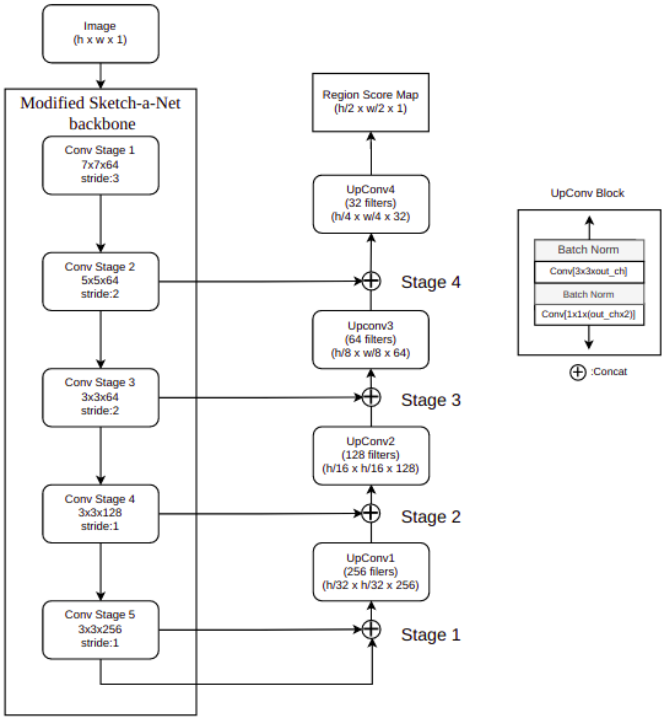

Figure 4.6: Architecture of modified CRAFT based on [3]

(a) Replaced 15x15 convolutions of the first layer of Sketch-a-Net [57] with 5x5 convolutions.

(b) Increased the stride of convolutions and removed max-pooling blocks of the backbone.

2. The regressor block of CRAFT is removed and the output of the last upconv block is encoded as the regional score.

The above model was trained with MSE loss and we obtain axis aligned bounding boxes with using a modified watershed algorithm

Further for image based models, we also train appropriately two state-of-the-art generic object detectors – *DSOD [44]* and *Tiny-YOLOv4* [52]. We also train modified versions of popular scene text detection models – *TextBoxes++ [27]*.

4.4 Experiments and Results

We first describe the experiments and results for atypical content detection. Following standard machine learning protocols, we divide data into training, validation and test splits. For each target phrase, the sessions containing atypical content are randomly split in the ratio 70 (train) : 15 (validation) : 15 (test). Since we perform data augmentation on atypical content-free sessions, we divide such

sessions in the aforementioned ratio as well. The respective data splits are combined to obtain the final groups.

4.4.1 Atypical content detection

Evaluation Protocol: We conduct evaluation using two protocols. In the first protocol, we consider models trained to detect all atypical content classes. To score performance, we use the standard object detection measures – mean-average-precision (mAP) and mean-average-recall (mAR) [59]. These measures are typically reported on a $[0, 1]$ scale – larger the better. mAP and mAR are reported at an Intersection-over-Union (IoU) threshold of 0.5. In other words, an overlap of 50% or greater between predicted bounding box and ground-truth bounding box is deemed correct (assuming predicted category label also matches).

Training: For training CANVASNET, we employ Adam optimizer [25] with a mini-batch size of 8, the exponential decay rate for $1st$ and $2nd$ moment estimates set to 0.9 and 0.999 respectively. We stop training after 50 epochs. Classification loss weight (α) is set to 1000 for quick convergence. The training takes approximately 4.5 hours on two GTX 1080Ti 11GB GPUs. We use *mish*[33] activation function for both text-only and multiclass models. As per our experiments, *mish* activation leads to fast convergence, as well as slight improvement is observed in mAP, mAR as compared to training using *relu* activation. Detecting smaller objects is a harder task than detecting object instances with larger sizes. As per[46], smaller objects are considered harder to learn than larger objects. Hence, we adopt a curriculum learning[4] strategy to gradually train the network from *easy samples* (large objects) to *hard samples* (small objects). We divide the training dataset into three sets: S_{easy}, S_{medium}, and S_{hard}, based on the bounding box area. For an individual task $T_i, i \in 1, 2, 3$, we train the detection model in three steps that can be formulated as:

$$T_i = \begin{cases} S_{\text{easy}} & I_1;\ \text{if } Ar_{bbox} < Ar_{\text{easy}} \\ S_{\text{easy}} + S_{\text{medium}} & I_2;\ \text{if } Ar_{bbox} < Ar_{\text{medium}} \\ S_{\text{easy}} + S_{\text{medium}} + S_{\text{hard}} & I_3 \end{cases} \tag{4.2}$$

I_1, I_2, and I_3 are the number of iterations each group of the sets are trained. Ar_{easy} and Ar_{medium} are the area thresholds for selecting large and medium bounding boxes. Further, for training text-only model, we also used *Hard mining training regime* for each S_{easy}, S_{medium}, and S_{hard} sets which involves mining true-positive samples having overlap threshold value with ground truth lies in window of length 0.1 around 0.5 i.e it lies in $[0.45, 0.55]$. We observed significant increase in mAP, mAR as compared to standard CANVASNET with regular training regime. For training multiclass model, we used mini-batch re-sampling to ascertain each class has equal probability of appearing in any training batch.

We observed that for some of our training examples, prediction boxes representing same atypical instance were overlapping to the extent of one lying completely inside other, and hence while evaluation

Table 4.3: CANVASNET performance for atypical content classes. IoU=0.5 refers to detection threshold. Refer to Sec. 4.4.1 for details.

Atypical Content Category	IoU=0.5	
	mAP	mAR
Text	0.58	0.80
Number	0.44	0.61
Icon	0.55	0.68
Circle	0.72	0.85

Table 4.4: Performance comparison with baselines. mAP = mean Average Precision, mAR = mean Average Recall, #Parameters = the number of trainable weights in the corresponding deep network in millions, ADT = average detection time per image in milliseconds.

Method	Text only		Multiclass		# Parameters	ADT
	mAP	mAR	mAP	mAR	M=million	(m.sec)
CANVASNET	**0.78**	**0.90**	**0.41**	**0.53**	1.90 M	35
BiLSTM+CRF [7]	0.06	0.04	0.02	0.03	0.01 M	85
SketchsegNet+[35]	0.56	0.32	0.04	0.11	3.90 M	21
Tiny-YOLOv4[52]	0.26	0.65	0.31	0.51	5.88 M	40
TextBoxes++[27]	0.41	0.65	0.25	0.39	29.31 M	51
DSOD[44]	0.43	0.66	0.25	0.40	17.49 M	52
CRAFT [3]	0.47	0.69	0.17	0.30	1.18 M	34

of CANVASNET, we merged such overlapping prediction boxes which improved our predictions qualitatively as well as increased mAP by a slight margin.

Detection Results: CANVASNET's performance for all the atypical categories can be viewed in Table 4.3. Fig. 4.7 depicts examples of CANVASNET detections, including some failure cases. We conducted ablation experiments to determine the relative importance of our architectural and optimization choices. Details of the ablation configurations and results can be viewed in the project page. The comparative evaluation of CANVASNET with baseline approaches is summarized in Table 4.4. Note that the comparison also includes compute-related aspects (number of trainable parameters in the approaches and average detection time).

Table 4.3 shows CANVASNET's performance for various atypical content categories. The consistent depictions of *Circle* enables good performance for the category. Detecting isolated numbers is slightly more challenging. Empirically, we observed that sketched content resembling letters (e.g. a mountain sketch) or numerals (e.g. vertical bar groups) accounted for most of the false positive detections. Text spanning a significantly large extent of the canvas and unusually oriented numbers accounted for majority of the missed detections (false negatives). Fig 4.8, Fig 4.9, Fig 4.10 and Fig 4.11 show qual-

Table 4.5: Performance scores for CANVASNET ablative variants (*Text*).

Ablation Type	Pipeline Component	Ablation Details	mAP	mAR
Architectural	Feature Extractor (Fig. 4.3)	Including output of SC Transition Layer-2 among prediction feature maps	0.64	0.78
		Reduce number of dense segments (-1)	0.46	0.77
		Increase number of dense segments (+1)	0.49	0.72
	Multi-scale Predictor (Fig. 4.3)	Reduce downsampling blocks (-1)	0.60	0.80
	Entire Model	Replace Separable convolution with regular convolution	0.42	0.66
Optimization	N/A	Cross entropy loss for classification	0.45	0.86
		L1 loss for regression	0.60	0.79
		$\alpha = 10$	0.25	0.57
		$\alpha = 100$	0.45	0.71
		CANVASNET	**0.78**	**0.90**

itative detection results of the CANVASNET model for atypical content category text, number, symbol and circle respectively. False negatives are shown as dashed rectangles and false positives as dotted rectangles.

We also conducted ablation experiments for the setting with *Text* as the atypical category to determine the relative importance of our architectural and optimization choices. The results can be seen in Table 4.5. In particular, our choice of using separable convolutions was a crucial decision, enabling a significant performance boost overall. The crucial nature of relative tradeoff α between the two components (classification, regression) in the loss function can also be seen in Table 4.5.

From Table 4.4, we see that CANVASNET clearly outperforms a variety of baseline approaches (Sec. 4.4.1). This is predominantly due to the carefully considered architectural and optimization choices in designing CANVASNET. The results also illustrate the superiority of image-based approaches compared to the sketch stroke processing approaches (*BiLSTM +CRF, SketchSegNet+*). Keeping the rule-violation detection scenario in mind, we also trained variants designed to detect the single class *Text*. As the 'Text only' column in Table 4.4 shows, CANVASNET remains the best performer. From the table (column named 'Parameters'), we also note that CANVASNET achieves its superior performance despite containing a smaller number of parameters relative to most of the baselines.

4.5 Summary

In this chapter, we described CANVASNET, a deep neural network to detect atypical sketch content in drawing canvas and then we compared CANVASNET with various baselines and its ablative variants. In the next chapter, we will discuss how we employ CANVASNET to design a distributed and scalable system for close to real time sketch content-based alert generation.

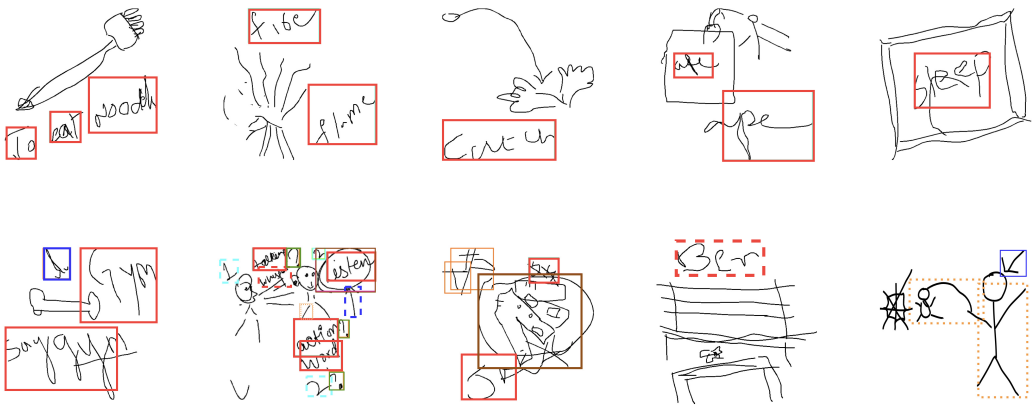

Figure 4.7: Examples of atypical content detection by CANVASNET. False negatives are shown as dashed rectangles and false positives as dotted rectangles.

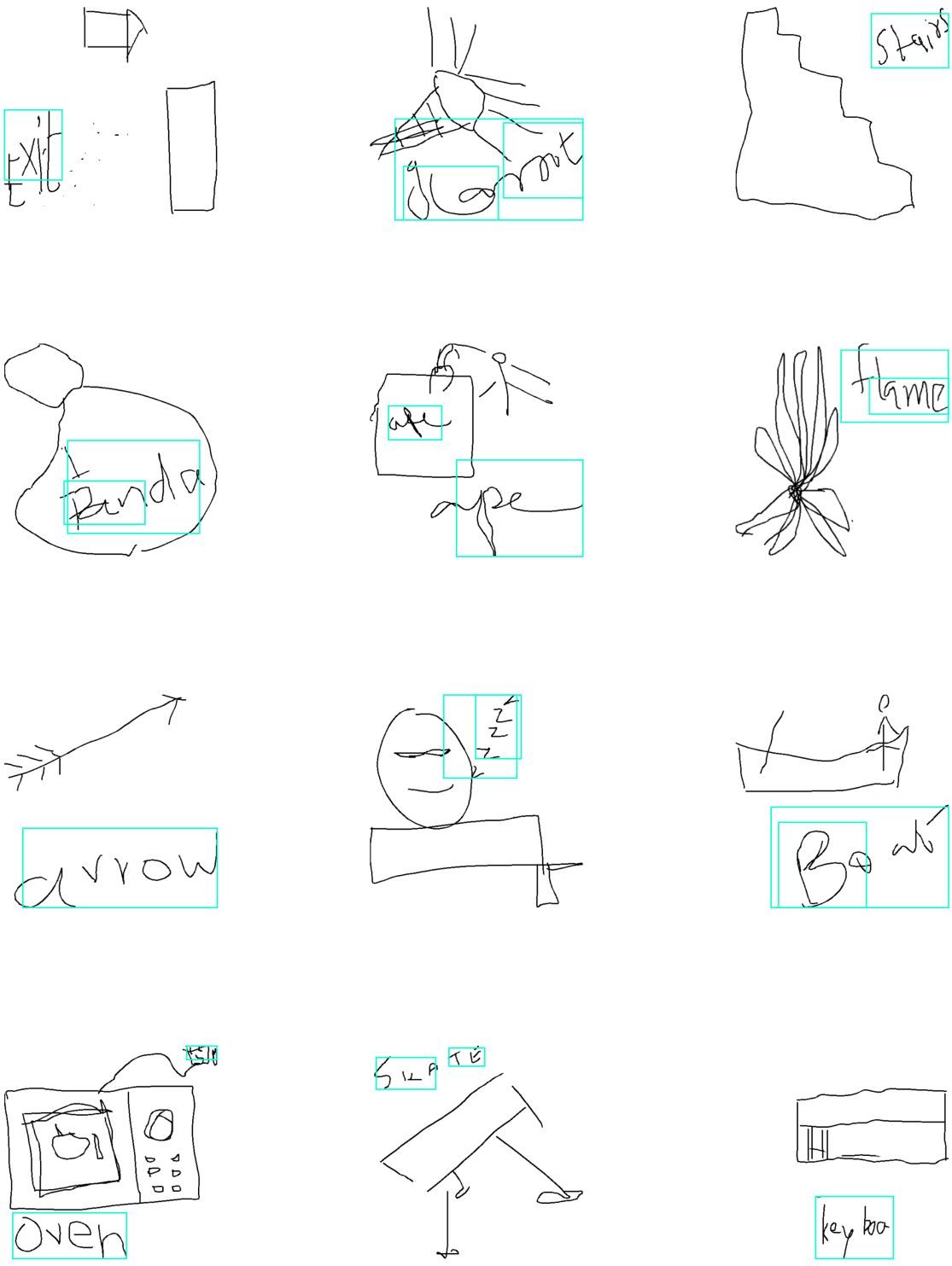

Figure 4.8: CANVASNET detections for Text class

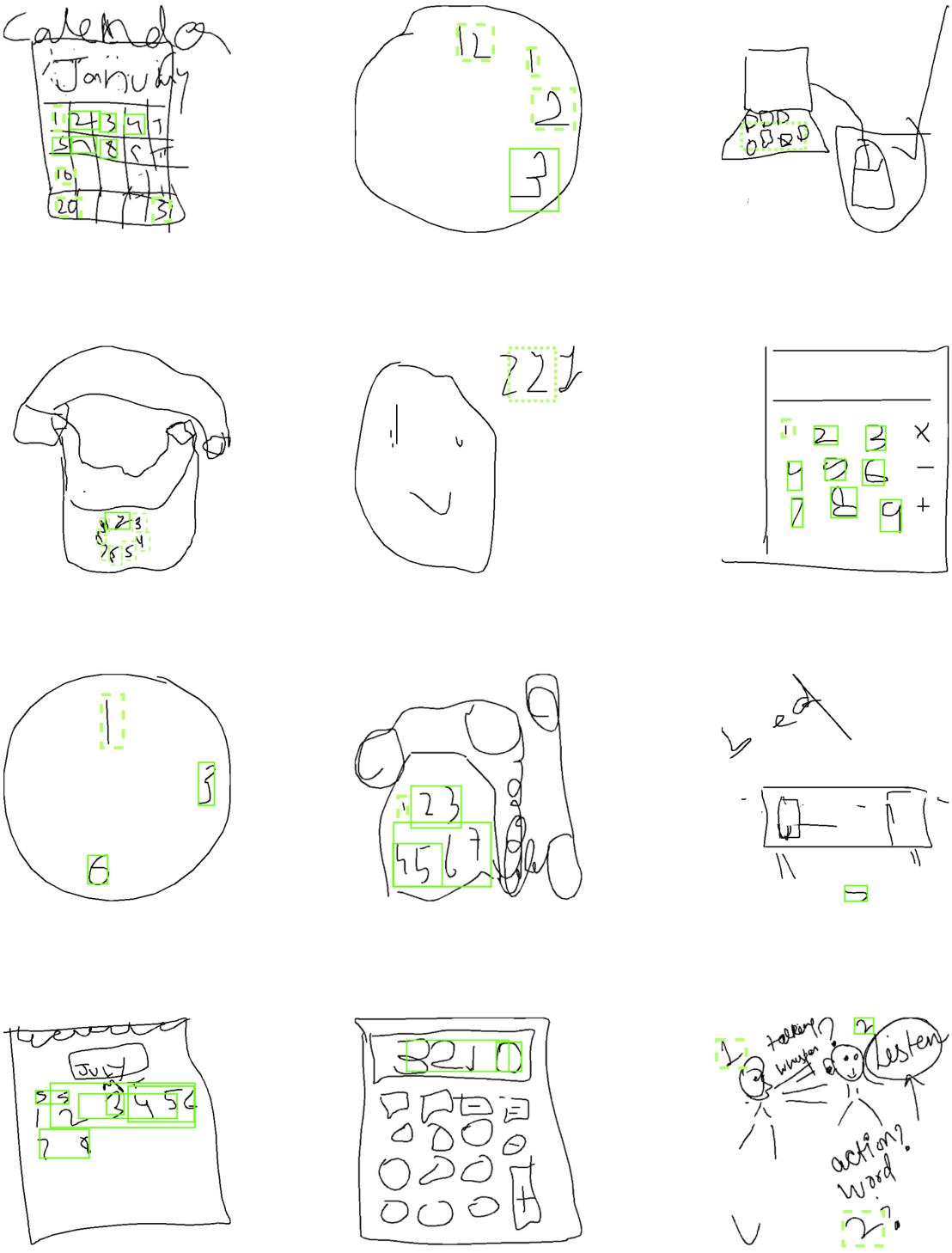

Figure 4.9: CANVASNET detections for Number class

Figure 4.10: CANVASNET detections for Icon class

Figure 4.11: CANVASNET detections for Circle class

Chapter 5

DRAWMON: A Distributed System for Sketch Content-based Alert Generation

5.1 Introduction

Consider a scenario with multiple online Pictionary game sessions in progress. We require a framework for automatic and concurrent monitoring of these game sessions for any atypical activities (e.g. a rule violation such as writing text on canvas). Such a framework needs to be reliable, scalable and time-efficient. To meet these requirements, we propose DRAWMON - a distributed alert generation system (see Fig. 5.1). DRAWMON monitors multiple online Pictionary game sessions to detect atypical activities and generates close to real time alerts for flagging anomalous activities. Each game session is managed by a central Session Manager which assigns a unique session id (Fig. 5.2). For a given session, whenever a sketch stroke is drawn, the accumulated canvas content (i.e. strokes rendered so far) is tagged with session id and relayed to a shared Session Canvas Queue. For efficiency, the canvas content is represented as a lightweight Scalable Vector Graphic (SVG) object. The contents of the Session Canvas Queue are dequeued and rendered into corresponding 512×512 binary images by Distributed Rendering Module in a distributed and parallel fashion. The rendered binary images tagged with session id are placed in the Rendered Image Queue. The contents of Rendered Image Queue are dequeued and processed by Distributed Detection Module. Each Detection module consists of our custom-designed deep neural network CANVASNET which processes the rendered image as input. CANVASNET outputs a list of atypical activities (if any) along with associated meta-information (atypical content category, 2-D spatial location).

The outputs from multiple distributed CANVASNET instances within the Distributed Detection Module are routed to the Alert Generator Module. An activity Record Table within this module records information related to ongoing game sessions and atypical content instances. This table is analyzed with respect to a Rule Base sub-module which generates appropriate alerts and relays them to the appropriate game sessions. Since rule violations are of predominant interest, other atypical content alerts

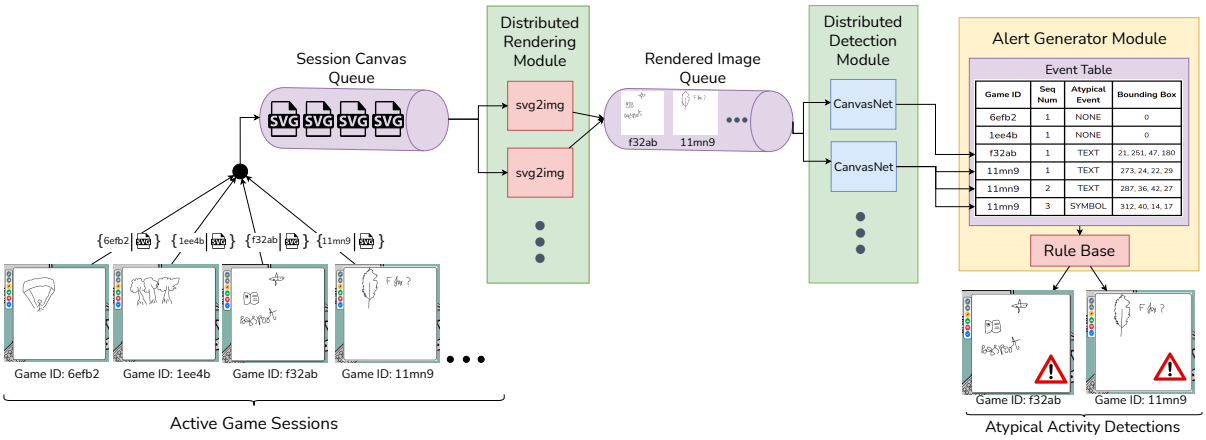

Figure 5.1: A pictorial overview of DRAWMON - our distributed atypical sketch content response system. Also see Fig. 5.2 for additional architectural details.

can be filtered out. Incoming alerts are finally displayed on the game session user interface (UI) - see Fig. 2.2.

Also note that two manually-controlled mechanisms related to alert generation exist within the game UI. The Guesser player can press a button labelled 'Drawer is violating rule!'. This simply generates the alert (but does not highlight the canvas location where violation occurs). On the Drawer's side, the button 'False Alarm' can be used to dismiss false positive alerts (see Fig. 2.2). In the current deployment, we utilize the *Text* detection variant of CANVASNET to detect text writing event on canvas.

5.2 DrawMon User Study Experiments

To quantify the efficacy of DRAWMON, we analyzed game session data with DRAWMON deployed to detect text. The canvas contents are relayed to DRAWMON every 1 second. We deployed 4 CANVASNET instances within the Distributed Detection Module on two 2080Ti GPUs alongside 16 worker processes for svg to image conversion. The combined peak usage of GPU RAM was 20 GB while peak CPU RAM usage was 15 GB. 23 participants (11 male, 12 female) in the age group $19-25$ (mean=20.8, std.=3.1), recruited using social media and from the institution's student pool, participated in the study. Each session had an average duration of 47.5 sec (std.= 37.4) with the maximum being 120 seconds. Over the study period, the maximum number of concurrent game sessions managed by DRAWMON was 4. From the resulting set of 145 game sessions, 69 contained atypical text activities. During the sessions, we recorded timestamped alerts from DRAWMON, false alarm notifications by the Drawer player and rule violation notifications from the Guesser player. The results from the study are summarized in Table 5.1. To determine DRAWMON's throughput, we measured two quantities. The first, processing time (`p-time`), is the average elapsed time between the canvas representation being sent to DRAWMON and receiving an alert. In case no alerts were generated, the timestamp corresponding to end of CANVASNET processing was considered. From our data, `p-time` was 0.4s. The other measurement

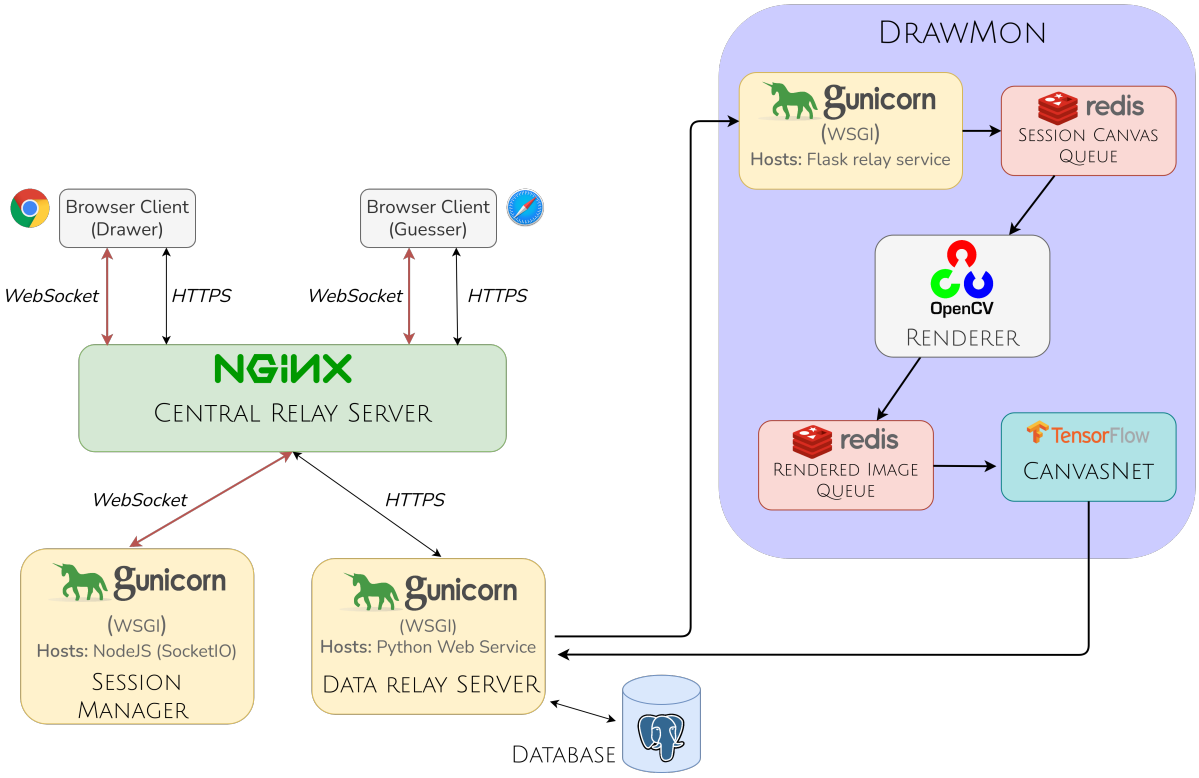

Figure 5.2: System architecture for Pictionary game setup and DRAWMON.

Table 5.1: User study statistics with DRAWMON deployed.

Game Event Type	Count
(True Positive) DRAWMON generates 'Rule Violation' alert. Drawer doesn't press 'False Alarm' button.	62
(False Positive) DRAWMON generates 'Rule Violation' alert. Drawer presses 'False Alarm' button.	32
(False Negative) No 'Rule Violation' alert. Guesser presses 'Drawer is violating rule' button.	6

was the maximum number of concurrently active sessions (`n-sess`) – this was 4. Defining the effective throughput rate to be `tpr` = `p-time`/`n-sess`, we obtain an average processing rate of 10 items per second.

Results and Analysis: The results from DRAWMON deployment user study are summarized in Table 5.1. From the table, we see that a significant fraction of DRAWMON generated alerts are valid (see 'True Positives'). From the results, DRAWMON's precision is 0.66 while recall is 0.91. Post the user study, we conducted a brief survey with Likert-type questions on a 1 to 5 scale with 5 being the best. 'Q: How responsive was DRAWMON to valid rule violations?' : The average score was 3.63 (s.d.=0.74), indicating reasonably high system throughput despite multiple concurrent sessions. This is also supported by the recorded throughput rate (`tpr`) mentioned previously in this section. 'Q: How was the overall game experience?' : The score was 3.91 (s.d.=0.60), suggesting a positive session experience and satisfaction with rule violation detection and response mechanisms.

Table 5.2: DRAWMON throughput comparison while keeping system resources fixed and increasing the number of concurrent virtual sessions.

Total Sessions	Atypical Event Sessions	Effective Throughput
145	69	40 fps
200	94	40 fps
300	128	40 fps
400	191	40 fps
500	263	40 fps
1000	509	40 fps

Table 5.3: DRAWMON throughput comparison on varying system resources for a fixed set of 1000 game sessions.

CANVASNET Instances	Worker Processes	Effective Throughput
1	16	21 fps
1	12	26 fps
2	6	35 fps
2	12	40 fps

5.3 Scalability Studies and Results

We conducted a simulation study on scalability and performance of DRAWMON. In the DRAWMON system (see Fig. 2.2), a recorded game session's Drawer and Guesser events can be played back by assigning virtual players and mimicking the actual game. Taking advantage of this capability, we created virtual Drawer and Guesser players from 'test set' game sessions. Note that the ground-truth for atypical activities is available for the test set.

In the first study, we examined the effect of increasing the number of concurrent virtual sessions on DRAWMON throughput while keeping system resources fixed (i.e. 2 CANVASNET instances, 12 worker processes). The game sessions were selected randomly with replacement.

Results shown in Table- 5.2 indicate DRAWMON's ability to reasonably sustain the throughput rate and its scalability in handling larger number of concurrent sessions containing atypical events.

In the second study, we examined the effect of increasing the system resources for a fixed set of 1000 game sessions. As before, the virtual game sessions were selected randomly with replacement.

Results shown in Table- 5.3 indicate DRAWMON's ability to benefit from additional allocation of system resources to meet demand.

5.4 Application Scenarios

Application Scenarios: Although we have used Pictionary as a use case scenario, we expect *DrawMon* to be suitable for other shared and interactive whiteboard scenarios. For instance, in a writing related

setting, the notion of atypical categories can be the exact opposite of Pictionary scenario: text on canvas would be routine while drawings might be considered abnormal. This can be tackled by appropriate data labelling, for e.g. using our CANVASDASH annotation tool, and subsequently retraining CANVASNET deep network. In another scenario, consider participants grouped into teams for a collaborative scene drawing task [8, 55]. DRAWMON, using a CANVASNET configured for sketched scene recognition [61], can alert the instructor on progress and task completion. For this task, a CANVASNET instance trained to recognize individual objects and iconic components from our dataset (e.g. arrows, 'addition mark') could also be included as additional detection component for expanding the detection capability.

Chapter 6

Conclusions

In this thesis, we addressed the problem of detection of controversial content drawn by malicious participants in increasingly popular shared digital whiteboards in both educational and workplace settings for collaboration and communication. Such activities tend to impair the collective experience of participants and hence the broad requirement is for a framework which can respond to a variety of atypical whiteboard content in a reliable, comprehensive and timely manner. Taking Pictionary – a wonderful example of cooperative game play to achieve a shared goal in communication- restricted settings, as a use case we first collected ATYPICT – the first ever dataset of atypical whiteboard content, using our custom browser based 2 player Pictionary-like sketching game dubbed PICTGUESS. Borrowing terminology from the seminal work of von Ahn and Dabbish [51], Pictionary can be considered an 'inversion game' with full transparency. Riberio and Igarashi [37] employ a sketching-based interactive guessing game to progressively learn visual models of objects. A review of Pictionary-like word guessing games involving drawing can be found in the work by Sarvadevabhatla et al. [42]. In general, most of the existing works are confined to idealized toy settings with a small number of visually simple categories. Unlike what we proposed our work, they do not discuss the possibility of atypical activity.

We then finally introduced DRAWMON, a system for canvas activity-based alert generation. DRAWMON is a distributed framework for monitoring multiple shared interactive whiteboards for detecting atypical content. DRAWMON is enabled by a number of equally important lateral contributions - (i) CANVASDASH - an intuitive dashboard UI for annotation and visualization (ii) ATYPICT - a first of its kind dataset for atypical sketch content (iii) CANVASNET - a deep neural network for atypical content detection. Together, these reusable contributions create the possibility of developing similar frameworks for other shared and interactive whiteboard scenarios. Apart from atypical content detection, we expect our game session dataset to be a valuable resource in itself for analyzing player characteristics and strategies in communication restricted non-adversarial games [15].

Our findings and other project related details can be explored at https://drawm0n.github.io. The website includes links to code and pre-trained models for CANVASNET, DRAWMON, the new atypical sketch-content dataset ATYPICT introduced by us and some example annotation session, gameplay

session from our CANVASDASH - annotation tool and browser-based pictionary data collection tool for additional exploration and benefit of the community.

6.1 Future Work

The development of AI agents which can generate game moves is an active research area [47, 45]. In addition to training on human-vs-human game data, agents are also trained to play against themselves. The ability to recognize sketches forms a crucial capability for AI agents participating in Pictionary. In recent times, active research has been conducted on collecting sketches of objects [21] and scenes [61] in order to build recognizer agents [41], especially in toy, Pictionary-like settings [42]. Pictionary is complicated by additional factors such as multi-modal game play (guesser uses speech/lexical modality while drawer uses visual modality), asynchronous turn- taking and only high-level notions of what constitutes a 'win' (e.g. target word is 'airplane', but guesser's proposal of 'warplane' may be considered acceptable). With the sketch recognizer in place, we plan to develop practical AI agents which can mimic human Pictionary players in a more interactive, realistic manner compared to existing non-interactive works [5, 16].

Related Publications

Nikhil Bansal, Kartik Gupta, Kiruthika Kannan, Sivani Pentapati, Ravi Kiran Sarvadevabhatla. *DrawMon: A Distributed System for Detection of Atypical Sketch Content in Concurrent Pictionary Games*. Proceedings of the 30th ACM International Conference on Multimedia, 2852–2861 (2022). https://doi.org/10.1145/3503161.3547747

Bibliography

[1] S. Andolina, H. Schneider, J. Chan, K. Klouche, G. Jacucci, and S. Dow. Crowdboard: augmenting in-person idea generation with real-time crowds. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, pages 106–118, 2017.

[2] A. Awal, G. Feng, H. Mouchère, and C. Viard-Gaudin. First experiments on a new online handwritten flowchart database. In *Electronic Imaging*, 2011.

[3] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning.

[5] A. K. Bhunia, A. Das, U. Riaz Muhammad, Y. Yang, T. M. Hospedales, T. Xiang, Y. Gryaditskaya, and Y.-Z. Song. Pixelor: A competitive sketching ai agent. so you think you can beat me? In *SIGGRAPH Asia*, 2020.

[6] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[7] A. Darvishzadeh, T. Stahovich, A. Feghahati, N. Entezari, S. Gharghabi, R. Kanemaru, and C. Shelton. Cnn-blstm-crf network for semantic labeling of students' online handwritten assignments. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1035–1040. IEEE, 2019.

[8] S. M. Davidson, N. M. Benson, and S. R. Beach. Drawn together: a curriculum for art as a tool in training. *Academic Psychiatry*, 45(3):382–387, 2021.

[9] L. Deng, Y. Gong, Y. Lin, J. Shuai, X. Tu, Y. Zhang, Z. Ma, and M. Xie. Detecting multi-oriented text with corner-based region proposals. *Neurocomputing*, 334:134–142, Mar 2019.

[10] P. V. Dinh, T. N. Nguyen, and Q. U. Nguyen. An empirical study of anomaly detection in online games. In *2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)*, pages 171–176, 2016.

[11] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):44:1–44:10, 2012.

[12] E. Elyan, L. Jamieson, and A. Ali-Gombe. Deep learning for symbols detection and classification in engineering drawings. *Neural networks : the official journal of the International Neural Network Society*, 129:91–102, 2020.

[13] D. Fang, H. Xu, X. Yang, and M. Bian. An augmented reality-based method for remote collaborative real-time assistance: from a system perspective. *Mobile Networks and Applications*, pages 1–14, 2019.

[14] N. Fay, B. Walker, and N. Swoboda. Deconstructing social interaction: The complimentary roles of behaviour alignment and partner feedback to the creation of shared symbols. *Proceedings of the 39th Annual Meeting of the Cognitive Science Society*, pages 26–29, 2017.

[15] K. I. Gero, Z. Ashktorab, C. Dugan, Q. Pan, J. Johnson, W. Geyer, M. Ruiz, S. Miller, D. R. Millen, M. Campbell, S. Kumaravel, and W. Zhang. Mental models of ai agents in a cooperative game setting. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.

[16] F. Huang, E. Schoop, D. Ha, and J. Canny. Scones: towards conversational authoring of sketches. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 313–323, 2020.

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[18] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks, 2014.

[19] A. B. Jeng and C. L. Lee. A study on online game cheating and the effective defense. In M. Ali, T. Bosse, K. V. Hindriks, M. Hoogendoorn, C. M. Jonker, and J. Treur, editors, *Recent Trends in Applied Artificial Intelligence*, pages 518–527, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[20] M. M. Jensen, R. Rädle, C. N. Klokmose, and S. Bodker. Remediating a design tool: implications of digitizing sticky notes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[21] J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg. The quick, draw! ai experiment. *Mountain View, CA*, 2016.

[22] F. D. Julca-Aguilar and N. S. T. Hirata. Symbol detection in online handwritten graphics using faster r-cnn. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 151–156, 2018.

[23] K. Kaiyrbekov and M. Sezgin. Stroke-based sketched symbol reconstruction and segmentation, 2019.

[24] M. Kam, J. Wang, A. Iles, E. Tse, J. Chiu, D. Glaser, O. Tarshish, and J. Canny. Livenotes: A system for cooperative and augmented note-taking in lectures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 531–540, New York, NY, USA, 2005. Association for Computing Machinery.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] L. Li, H. Fu, and C.-L. Tai. Fast sketch segmentation and labeling with deep learning, 2018.

[27] M. Liao, B. Shi, and X. Bai. Textboxes++: A single-shot oriented scene text detector. *IEEE transactions on image processing*, 27(8):3676–3690, 2018.

[28] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[30] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[31] Y. Liu, S. Zhang, L. Jin, L. Xie, Y. Wu, and Z. Wang. Omnidirectional scene text detection with sequential-free box discretization, 2019.

[32] T. Matsushita and M. Nakagawa. A database of on-line handwritten mixed objects named "kondate". In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 369–374, 2014.

[33] D. Misra. Mish: A self regularized non-monotonic activation function, 2020.

[34] A. Perlich and C. Meinel. Cooperative note-taking in psychotherapy sessions: An evaluation of the therapist's user experience with tele-board med. In *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6. IEEE, 2018.

[35] Y. Qi and Z.-H. Tan. Sketchsegnet+: An end-to-end learning of rnn for multi-class sketch semantic segmentation. *IEEE Access*, 7:102717–102726, 2019.

[36] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.

[37] A. Ribeiro and T. Igarashi. Sketch-editing games: Human-machine communication, game theory and applications. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, page 287–298, New York, NY, USA, 2012. Association for Computing Machinery.

[38] M. J. Rogerson, M. R. Gibbs, and W. Smith. Cooperating to compete: The mutuality of cooperation and competition in boardgame play. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 1–13, New York, NY, USA, 2018. Association for Computing Machinery.

[39] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[40] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 2016.

[41] R. K. Sarvadevabhatla, J. Kundu, and V. B. R. Enabling my robot to play pictionary: Recurrent neural networks for sketch recognition. In *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, page 247–251, New York, NY, USA, 2016. Association for Computing Machinery.

[42] R. K. Sarvadevabhatla, S. Surya, T. Mittal, and R. V. Babu. Pictionary-style word guessing on hand-drawn object sketches: Dataset, analysis and deep network models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):221–231, 2020.

[43] B. Schäfer and H. Stuckenschmidt. Arrow r-cnn for flowchart recognition. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 1, pages 7–13. IEEE, 2019.

[44] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue. Dsod: Learning deeply supervised object detectors from scratch. In *Proceedings of the IEEE international conference on computer vision*, pages 1919–1927, 2017.

[45] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[46] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe. Curriculum self-paced learning for cross-domain object detection. *Computer Vision and Image Understanding*, 204:103166, 01 2021.

[47] G. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219, 1994.

[48] L. B. The and V. N. Khanh. Gameguard: A windows-based software architecture for protecting online games against hackers. In *Proceedings of the 2010 Symposium on Information and Communication Technology*, SoICT '10, page 171–178, New York, NY, USA, 2010. Association for Computing Machinery.

[49] J. Tunell. Classification of offensive game-emblem drawings using convolutional neural networks and transfer learning. Master's thesis, Uppsala University, 2018.

[50] T. Van Phan and M. Nakagawa. Text/non-text classification in online handwritten documents with recurrent neural networks. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 23–28. IEEE, 2014.

[51] L. von Ahn and L. Dabbish. Designing games with a purpose. *Commun. ACM*, 51(8):58–67, Aug. 2008.

[52] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Scaled-yolov4: Scaling cross stage partial network. *arXiv preprint arXiv:2011.08036*, 2020.

[53] F. Wang, S. Lin, H. Wu, H. Li, R. Wang, X. Luo, and X. He. Spfusionnet: Sketch segmentation using multi-modal data fusion. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1654–1659, 2019.

[54] W. Wang, E. Xie, X. Song, Y. Zang, W. Wang, T. Lu, G. Yu, and C. Shen. Efficient and accurate arbitrary-shaped text detection with pixel aggregation network, 2020.

[55] B. Williford, M. Runyon, W. Li, J. Linsey, and T. Hammond. Exploring the potential of an intelligent tutoring system for sketching fundamentals. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[56] L. Yang, J. Zhuang, H. Fu, K. Zhou, and Y. Zheng. Sketchgcn: Semantic sketch segmentation with graph convolutional networks, 2020.

[57] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales. Sketch-a-net that beats humans. In M. W. J. Xianghua Xie and G. K. L. Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 7.1–7.12. BMVA Press, September 2015.

[58] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12993–13000, 2020.

[59] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang. Iou loss for 2d/3d object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94. IEEE, 2019.

[60] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017.

[61] C. Zou, Q. Yu, R. Du, H. Mo, Y.-Z. Song, T. Xiang, C. Gao, B. Chen, and H. Zhang. Sketchyscene: Richly-annotated scene sketches. In *Proceedings of the european conference on computer vision (ECCV)*, pages 421–436, 2018.