Towards Efficient Semantic Segmentation Compression via Meta Pruning

Ashutosh Mishra[1*], Shyam Nandan Rai[2*], Girish Varma[1], and CV Jawahar[1]

[1] CVIT, KCIS, IIIT Hyderabad
[2] Politecnico di Torino
ashutosh.mishra@research.iiit.ac.in, {girish.varma,jawahar}@iiit.ac.in
shyamnandanraii@gmail.com

Abstract. Semantic segmentation provides a pixel-level understanding of an image essential for various scene-understanding vision tasks. However, semantic segmentation models demand significant computational resources during training and inference. These requirements pose a challenge in resource-constraint scenarios. To address this issue, we present a compression algorithm based on differentiable meta-pruning through hypernetwork: MPHyp. Our proposed method MPHyp utilizes hypernetworks that take latent vectors as input and output weight matrices for the segmentation model. L_1 sparsification follows the proximal gradient optimizer, updates the latent vectors and introduces sparsity leading to automatic model pruning. The proposed method offers the benefit of achieving controllable compression during the training and significantly reducing the training time. We compare our methodology with a popular pruning approach and demonstrate its efficacy by reducing the number of parameters and floating point operations while maintaining the mean Intersection over Union (mIoU) metric. We conduct experiments on two widely accepted semantic segmentation architectures: UNet and ERFNet. Our experiments and ablation study demonstrate the effectiveness of our proposed methodology by achieving efficient and reasonable segmentation results.

Keywords: Semantic segmentation · Hypernetworks · Pruning · Compression.

1 Introduction

Semantic segmentation [2, 3, 23, 37] is a dense prediction task of assigning class labels to each pixel in an image. It has been widely used in autonomous driving, medical imaging, and satellite imaging applications. However, semantic segmentation models such as DeepLabV1,V2 [2, 3], DRN [37] generally have a large number of parameters. So deploying such models in environments with limited resources, such as mobile phones or embedded systems, can be challenging due to their large memory requirements during inference.

One potential approach to overcome the computational limitations is to adopt methods that reduce the model's complexity, like model compression, for instance, model pruning [7, 14, 24], or neural architecture search [8, 32, 34, 40].

* equal contribution

However, these methods above have their inherent problems. For instance, the method [32] is based on neural architecture search to find the best architecture within a predefined search space. The search space includes various architectural components, such as the number of layers, layer types (convolutional, recurrent, etc.), skip connections and other hyperparameters. However, it takes 8 Nvidia Turing GPUs with 24GB of VRAM to find such architectures, which is extremely expensive in terms of training complexity. On the other hand, pruning methods such as the lottery ticket hypothesis [7], a compression algorithm based on iterative pruning have shown a significant segmentation accuracy drop.

In order to efficiently prune semantic segmentation architectures, we propose a compression method based on differentiable meta pruning using hypernetworks MPHyp that substantially minimizes the training resources and simultaneously retains satisfactory segmentation performance. Our proposed technique is inspired from [19]. We call it meta-pruning since the parameters of the hypernetwork are responsible for generating weights of the segmentation model falling under the umbrella of meta-learning. The key concept behind the proposed approach is the hypernetwork that associates latent vectors for each layer in the segmentation network. This latent vector controls the output channel of the layer in consideration. Given the association of network layers, the latent vector serves as a controlling factor for the subsequent layer's input channel as well. During training, the hypernetwork receives the latent vectors from the current and preceding layers, which dictate the output and input channels of the current layer, respectively. The hypernetwork generates a weight matrix for the specific layer of the segmentation model. To achieve automatic pruning, we use the l_1 regularizer that helps in sparsification of the latent vectors. Subsequently, we employ proximal gradient algorithm to update and obtain the sparsified latent vectors. Together, this strategy leads to differentiability in the pruning mechanism. Once the compression ratio reaches the pre-determined level, the compression method stops. The sparsification of the latent vectors results in compressed outputs from the hypernetworks since the latent vectors and layers of the segmentation model are correlated. The proposed method offers the advantage of streamlining network pruning by focusing solely on the latent vectors, eliminating the need for additional complexities or human assistance. We find that our proposed method outperforms the baseline pruning method by a significant margin. We refrain from comparing with neural architecture search methods as they require huge training resources. To showcase the efficacy of our method, we performed extensive experimentation on IDD Lite [26]: a semantic segmentation dataset targeted for resource-constraint scenarios. In this context, resource-constraint means a lack of availability of better computing power. The images provided in this dataset are sampled and scaled from IDD [35], which are very different compared to other sophisticated semantic segmentation datasets like Cityscapes [5], Mapillary [28]. We chose UNet [31] and ERFNet [30] as our semantic segmentation models due to the wide acceptability and application of these networks in various domains [21, 25].

The contribution of our work is summarized below:

1. We propose an efficient semantic segmentation pruning method MPHyp based on hypernetwork. Our proposed method preserves significant segmentation performance after pruning and efficiently trains while requiring minimal training resources (See Section 3).
2. We compare our method with the baseline method and a popular pruning algorithm on UNet and ERFNet architectures trained on the IDD Lite dataset (See Section 4).
3. We perform ablation studies and experimentation to show the efficacy of our method (See Section 6).

2 Related Work

2.1 Semantic Segmentation

Semantic segmentation methods [11, 12] before deep learning utilize image features to perform segmentation. Fully Convolution Networks(FCN) [23] was the first deep learning-based method to output per pixel dense correspondences using a classification backbone for varying resolutions. The significant semantic segmentation performance gain of FCNs compared to non deep learning methods was due to the incorporation of skip connections between initial and final layers that combined coarse high features with fine low-level features. Subsequently, architectures such as Deeplabv1,Deeplabv2 [2, 39] were proposed that improved semantic segmentation performance by a) using post-processing strategies, such as Conditional Random Fields (CRF) to refine the output, b) replacing normal convolutions with dilated convolutions to enlarge the receptive field to gather more context but at the cost of increased computational overhead [2, 37], and c) increasing the field of view of the segmentation kernel by using features obtained from different strides and then aggregating them with average pooling and subsequent convolutional layers [13, 38].

Concurrently, [3, 4] proposed to use dilated convolutions inside the spatial pyramid pooling architecture to improve the accuracy along with increased training and inference time. Additionally, training the model on multiple scales and orientations of images, combined through pooling operations, was suggested for further accuracy improvement. Another work proposed attention-based methods to model long-range dependencies as well [15].

2.2 Semantic Segmentation Compression

To reduce the computation overload, neural architecture search (NAS) based methods have been adopted. It also helps to find lightweight models. Given a search space, neural architecture search-based methods help find a superior model. An example of NAS in semantic segmentation is [20]. This approach tries to find a repeatable cell structure along with the network architecture to improve the performance. However, it costs a huge amount of memory to reach the target architecture.

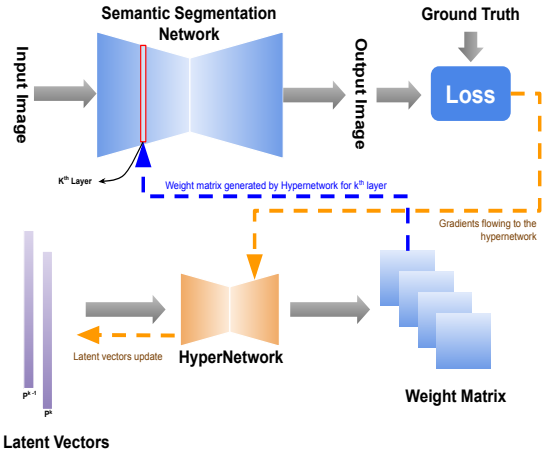

Fig. 1. Shows a pictorial representation of the proposed algorithm designed for compressing semantic segmentation networks using automatic differentiable pruning. Each layer of the segmentation network is associated with a latent vector. This latent vector is responsible for generating weights of the specific layer. While training, latent vectors get sparsified, leading to the pruning of the weights of the specific layer. The construction of the algorithm is such that the latent vector is covariant with its corresponding weight matrix. Weights produced by the hypernetwork generates the final output through the segmentation model.

A lot of compression methods tailoring to pruning have been proposed in the past. Architecture pruning, in general, aims to remove redundancies in the over-parameterized models for faster inference while maintaining most model non-redundant parameters. For instance, [10, 27] tries to compress models using specific hardware architecture or library support. This is referred to as non-structured pruning. On the other hand, some approaches perform pruning on the entire architecture [14, 24]. However, the performance of these pruning methods for complex tasks such as semantic segmentation has not been discussed.

2.3 HyperNetworks

Hypernetworks were first proposed in [9] that used a smaller network to generate weights for a bigger network. Hypernetworks have been widely used for neural architecture search [1], temporal forecasting [29], and also model compression [22]. The key idea behind hypernetwork is based on evolutionary methods that evolve smaller networks to generate weights of a larger network. A more efficient method to generate during the weight generation process, the search space is constrained within a smaller weight space [33]. Alternatively, the structure of the network can be kept fixed while weights are evolved through discrete cosine transform. This technique is referred to as a compressed weight search [18]. Similar to the aforementioned approach are Differentiable Pattern Producing

Networks (DPPNs) that evolve the structure of the network while the weight parameters are learned [6]. Apart from evolutionary-based algorithms, CNN-based approaches such as [16] generate weights to apply a transformation to the images. [17] uses the concept of dynamic filters that are generated based on input for the task of spatial transformation and deblurring.

As discussed in prior related works, segmentation models are generally large, and compression techniques such as neural architecture search require extensive computational resources to extract efficient semantic segmentation models. Our proposed method overcomes the aforementioned problem by leveraging hypernetworks that require minimal training resources.

3 Methodology

In this section, we discuss the process of compression of semantic segmentation models using hypernetwork. This is a lossy compression technique since the pruned connections cannot be restored due to the removal of redundant weight connections. Figure 1 illustrates the overall design of the proposed method, which is discussed in subsequent sections in more detail.

3.1 Architectural Design

We now describe the process of adapting hypernetwork in a semantic segmentation network. The proposed hypernetwork architecture constructed for a single layer neural network has the following layers:

1. ***Latent Layer***: that takes latent vectors as input and generates latent matrix.
2. ***Embedding Layer***: is responsible for projecting latent matrix to embedding matrix.
3. ***Output Layer***: transforms the embedding matrix to a final weight matrix of the corresponding layer.

As described above, we know the process of adapting single-layer neural network to hypernetwork, so we extend the same process for a semantic segmentation model. Suppose, we have a semantic segmentation model of K-layered network that requires to be pruned. Initially, we introduce a latent vector to all the layers that need to be pruned, as latent vectors are responsible for generating pruned weights. We keep the shape of the latent vector equal to the output channels for all the pruned layers to keep dimension consistency.

Assume for a given semantic segmentation layer k, the dimensionality of the weight matrix generated by k^{th} layer is $l{\times}m{\times}w{\times}h$, where l and m denote the output and input channels of k^{th} semantic segmentation layer, and $w{\times}h$ denotes its corresponding kernel width and height. Also, consider the latent vector for the k^{th} layer to be p^k. Since the size of the latent vector is equal to the number of output channels, the latent vector for k^{th} layer is $p^k \in R^l$. Thus, the dimensionality of the latent vector of the $k\text{-}1^{th}$ layer is $p^{k-1} \in R^m$.

Now, we pass latent vectors p^k and p^{k-1} as input to the hypernetwork corresponding to the k^{th} layer to generate the latent matrix, given by,

$$\mathbf{P}^k = \mathbf{p}^k.\mathbf{p}^{k-1^T} + \mathbf{V}_0^k, \tag{1}$$

where,

$$\mathbf{P}^k, \mathbf{V}_0^k \in R^{n \times c}$$

$[T]$ represents the transpose of the matrix and [.] denotes matrix multiplication. $\mathbf{V}_0$ is the bias matrix. Consequently, the latent matrix is further projected to an embedding dimension with the help of the embedding layer that is given by,

$$\mathbf{C}_{ij}^k = \mathbf{P}_{ij}^k.\mathbf{w}_1^k + \mathbf{v}_1^k \quad i = 1..l, j = 1...m, \tag{2}$$

$\mathbf{C}_{ij}^k, \mathbf{w}_1^k, \mathbf{v}_1^k \in R^e$, where $\boldsymbol{e}$ represents the embedding dimension. The elements of $\mathbf{w}_1^k$ and $\mathbf{v}_1^k$ are considered to be unique. We exclude the subscript *(i,j)* for ease of understanding. Now, we multiply $\mathbf{w}_1^k$, $\mathbf{v}_1^k$ and $\mathbf{P}_{ij}^k$ together to form a 3D matrix, given as $\mathbf{W}_1^k$, $\mathbf{V}_1^k$ and $\mathbf{C}_1^k \in R^{l \times m \times e}$ as formulated in eq. (2). We generate the final output $\mathbf{G}_{ij}^k$ by passing the embedding matrix eq. (2) through the output layer eq. (3).

$$\mathbf{G}_{ij}^k = \mathbf{C}_{ij}^k.\mathbf{w}_2^k + \mathbf{v}_2^k \quad i = 1..l, j = 1...m, \tag{3}$$

where,

$$\mathbf{G}_{ij}^k, \mathbf{v}_2^k \in R^{wh}$$

and,

$$\mathbf{w}_2^k \in R^{wh \times e}$$

We can observe the vectors $\mathbf{w}_2^k$, $\mathbf{v}_2^k$ and $\mathbf{G}_{ij}^k$ generate the final weight of the segmentation layer *k*. The final dimensionality of variables involved in eq. (3) are $\mathbf{W}_2^k \in R^{l \times m \times wh \times e}$ and $\mathbf{V}_2^k$ and $\mathbf{G}^k \in R^{l \times m \times wh}$.

In functional form, we can represent the output $\mathbf{G}^k$ as:

$$\mathbf{G}^k = h(\mathbf{p}^k, \mathbf{p}^{k-1}; \mathbf{W}^k, \mathbf{V}^k), \tag{4}$$

Here, h(.) indicates the overall transformation applied on the input latent vectors, parameterized by latent vectors and per layer weight tensors. In case of skip or residual connections present in the segmentation network, we handle it by concatenating the latent vector of the current layer and the associated skip connection layer to create the latent matrix using eq. (1) and eq. (3).

3.2 Sparsification of Latent Vectors

The next step of our proposed method is to introduce sparsification in latent vectors that are associated with layers of the semantic segmentation model. We introduce sparsity in latent vectors because all the latent vectors are connected to each other, and each latent vector is covariant with the output channels of the specific layer of the segmentation network. Sparsification of latent vectors

leads to a reduction in the count of output channels, leading to compression of the network. We achieve this by constraining latent vectors under L_1-norm formulated as:

$$R(p) = \sum_{k=1}^{K} ||p^k||_1 \tag{5}$$

where R is the regularization term. Now, we introduce differentiability with the help of the proximal gradient algorithm. In summary, the task of the proximal gradient algorithm is to sparsify and update the latent vectors. The latent vectors are updated using the proximal gradient algorithm given by,

$$\mathbf{p}[k+1] = \mathbf{prox}_{\lambda\mu R}(p[k] - \lambda\mu\nabla L(p[k])) \tag{6}$$

3.3 Semantic Segmentation Network Pruning

After the sparsification of latent vectors by employing the L_1-norm followed by proximal operation, the latent vectors are forced to become sparse without any human assistance, leading to automatic pruning. The latent vector sparsification leads to the generation of pruned weights for the corresponding layer of the segmentation network. After the compression stage, we obtain sparse latent vectors p^k and p^{k-1} for k^{th} semantic segmentation layer. After this, we apply masks t^k, t^{k-1} on the pruned vectors that are near zero with a predefined threshold τ. If the value is greater than the threshold, the returned value is one else zero. The sparsified latent vector, $\bar{p}^k$ is pruned using mask (t^k).

After the desired compression ratio is met, we then finetune the pruned semantic segmentation network to improve the accuracy. We refer finetuning as converging stage. The hypernetwork defined earlier gets scrapped, followed by the onset of the normal training regime. It is important to note that if the compression ratio increases, then the number of compression epochs also increases.

4 Experiments

4.1 Models and Dataset

We employ UNet [31] and ERFNet [30] as the base semantic segmentation architectures on which all the experiments are performed. We use IDD Lite [26] for our experiments. IDD Lite is a custom dataset specifically designed for resource-constrained scenarios. This dataset contains 1400 training images and 400 validation images sampled and scaled from the updated IDD dataset [35], keeping the largest dimension to 320 while preserving the image's aspect ratio with the hierarchy of 7 coarse labels.

4.2 Training Detials

We use a single NVIDIA 1080Ti GPU for simultaneous training and compression through our proposed method using the Pytorch deep learning library. The initial learning rate is set to 0.1, and total epochs are set to 250. We employ a reduced learning rate on the plateau scheduler. The training batch size is kept at 32, and the validation batch size is 4. The embedding dimension is fixed to 8 in all the experiments. The sparsity regularization factor is fixed to be 0.5.

4.3 Performance Metric

We evaluate the performance of the pruned semantic segmentation model through mean Intersection over Union metric, given as:

$$IOU = \frac{TP}{TP + FP + FN} \tag{7}$$

Here, TP, FP and FN are the number of true positives, false positives, and false negatives at the pixel level. For assessing the efficiency of pruning methods, we use the number of (a) floating point operations (GFlops), (b) parameters, and (c) flop ratio indicating the ratio of remaining flops and original flops.

Table 1. Shows the performance of our proposed method MPHyp on UNet and ERFNet. It can be observed that our method shows better performance than L1-pruning, with better accuracy and having a smaller network.

UNet					
Compression Ratio	Method	mIoU	GFlops	Params(M)	Flop Ratio
0.00	Un-Pruned	65.71	20.84	7.786	-
0.50	L1-Pruning [36]	43.18	10.42	3.897	0.5
	MPHyp	55.03	10.73	3.504	0.514
0.75	L1-Pruning [36]	26.71	5.212	1.951	0.25
	MPHyp	57.01	5.369	0.702	0.257
0.90	L1-Pruning [36]	19.09	2.098	0.778	0.10
	MPHyp	54.61	2.298	1.015	0.1102
0.95	L1-Pruning [36]	18.93	1.042	0.406	0.05
	MPHyp	52.9	1.361	0.76	0.065
ERFNet					
Compression Ratio	Method	mIoU	GFlops	Params(M)	Flop Ratio
0.00	Un-Pruned	58.06	3.72	2.063	-
0.50	L1-Pruning [36]	56.52	1.867	0.993	0.5
	MPHyp	55.49	1.794	0.959	0.482
0.75	L1-Pruning [36]	53.70	0.931	0.499	0.250
	MPHyp	55.14	0.937	0.408	0.251
0.90	L1-Pruning [36]	51.64	0.368	0.21	0.098
	MPHyp	52.24	0.403	0.128	0.118
0.95	L1-Pruning [36]	41.42	0.183	0.117	0.049
	MPHyp	51.63	0.256	0.109	0.068

5 Results

Quantitative Results: In this section, we discuss the results obtained from our proposed method: Meta-pruning based Hypernetwork (MPHyp) on UNet and ERFNet.

Table 1 shows the performance of MPHyp with L1-based channel pruning. We can observe that MPHyp preserves the segmentation accuracy at different compression ratios, whereas L1-pruning shows a significant drop in mIoU. Also, from the table, it is evident that for higher compression ratios, our proposed approach outperforms the L1-pruning technique, further decreasing the flops and parameter count. Simultaneously, MPHyp achieves a lower flop ratio and GFlops, resulting in faster network inference. We also show class-wise mIoU results for various pruning methods table 2 at different compression ratios. It

Table 2. Shows the mIoU performance of MPHyp at various compression ratios for UNet architecture. We can observe that MPHyp significantly outperforms L1-Pruning with higher mIoU at a higher compression ratio and shows comparable performance with the Un-Pruned network. It is also important to note that MPHyp shows better performance (in Bold) than the un-pruned method on the Vehicles and Roadside-Objects class, which could be advantageous in autonomous applications.

Methods	Drivable	Non-drivable	Living Thing	Vehicles	Roadside-Objects	Far-objects	Sky	mIoU
Compression Ratio:50 %								
Un-Pruned	92.51	36.89	49.97	69.79	43.19	72.96	94.67	65.71
L1-Pruning [36]	91.1	4.07	47.03	65.87	0.0	0.0	94.15	43.18
MPHyp	91.2	31.42	43.39	67.46	41.29	71.78	93.69	55.03
Compression Ratio:75 %								
Un-Pruned	92.51	36.89	49.97	69.79	43.19	72.96	94.67	65.71
L1-Pruning [36]	90.16	3.34	0.0	0.0	0.0	0.0	93.5	26.71
MPHyp	92.61	33.26	47.06	**72.28**	**44.27**	72.72	93.92	57.01
Compression Ratio:90 %								
Un-Pruned	92.51	36.89	49.97	69.79	43.19	72.96	94.67	65.71
L1-Pruning [36]	40.12	0.00	0.00	0.00	0.00	0.00	93.51	19.09
MPHyp	91.61	28.87	43.48	67.07	41.93	70.38	93.51	54.61
Compression Ratio:95 %								
Un-Pruned	92.51	36.89	49.97	69.79	43.19	72.96	94.67	65.71
L1-Pruning [36]	40.15	0.00	0.00	0.00	0.00	0.00	92.36	18.93
MPHyp	91.06	22.34	41.64	66.00	40.34	68.86	92.97	52.90

Table 3. Shows the MPHyp performance at various embedding dimensions. We can observe as the embedding dimension increases, the semantic segmentation decreases. However, on the other hand, we have improved the flop ratio with increasing embedding dimension. We keep the pruning ratio fixed at 0.5.

Method	Embedding Dimension	mIoU	Flop Ratio
MPHyp	8	55.03	0.5114
MPHyp	16	54.22	0.5196
MPHyp	32	53.82	0.5042

Table 4. Shows the MPHyp performance for different sparsity regularization formulations at a pruning ratio of 0.5.

Method	Proximal Regularization	mIoU	Flop Ratio
MPHyp	L1	55.03	0.5114
MPHyp	L2	53.81	0.5293

can be observed that MPHyp significantly outperforms L1-Pruning with higher mIoU at a higher compression ratio and shows comparable performance with an Un-Pruned network. Interestingly, MPHyp shows better performance than the un-pruned method on the Vehicles and Roadside-Objects class, which are important classes in autonomous applications. On average, the training time of MPHyp at different compression ratios was around 1Hr on 1080ti, which is significantly lower than existing neural architecture search methods and L1-pruning, which has an average training time of 3Hrs.

Qualitative Results: Figure 2 displays the qualitative results obtained from our proposed approach and the baseline pruning algorithm. The first row demon-

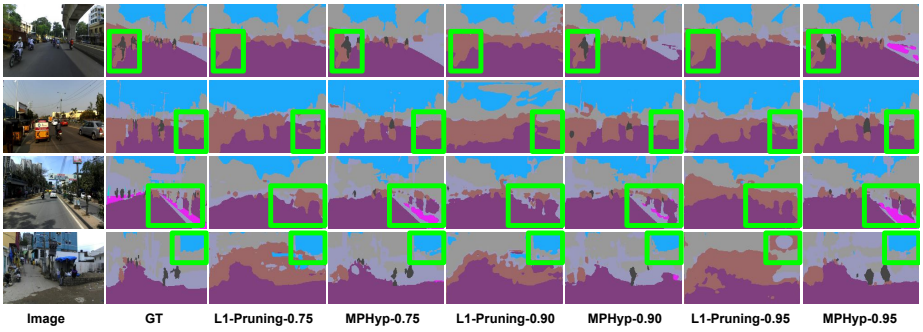

Fig. 2. Qualitative results at higher compression ratios 0.75, 0.9, and 0.95 comparing L1-pruning [36] and our proposed method is based on UNet architecture. Segmentation results enclosed in green boxes show that MPHyp is able to preserve segmentation results at higher pruning rates for different classes. (*Best viewed when zoomed*).

strates the prediction of classes *rider* and *bike*, and the second row focuses on the model's prediction for *car* class. The third and fourth rows illustrate classes *sidewalk* and *sky*, respectively. As it can be observed, MYHyp predictions have better fine details compared to L_1-pruning.

MYHyp-0.75 and MYHyp-0.90 predictions have finer details compared to MYHyp-0.95. L_1-pruning based approach's performance has very coarse boundaries for the objects of interest.

6 Ablations

We perform two ablations to see the performance of our proposed approach.

Embedding dimension : We increase the embedding dimension from the default value of 8 to 16 and 32 in table 3. We see that there is a trade-off between the mIoU value and the flop ratio. Though the flop ratio is marginally better for the higher embedding dimension, the mIoU metric follows a downward trend.

Proximal Gradient Regularizer : Now, we compare the effect of *L2* regularizer and *L1* regularizer to sparsify the latent vector, keeping the same pruning strategy in table 4. We can observe that *L1* regularizer shows better mIoU and flop ratio compared to *L2* regularizer. It is also important to note that to have a reasonable convergence speed λ used for *L2* regularizer must be significantly larger than that *L1* regularizer.

7 Conclusion

In this paper, we propose an automatic pruning method with the added advantage of differentiability with the help of hypernetworks for semantic segmentation architectures. The proximal gradient algorithm and L1 sparsification, along with the proposed hypernetwork design, help to find the compact representation of the architecture in hand. We use two widely accepted semantic segmentation architectures to show the effectiveness of our proposed approach at different

compression ratios. Our method opens a new direction of research toward an efficient pruning method for semantic segmentation.

Acknowledgment: This work has been partly supported by IHub-Data, Mobility at IIIT Hyderabad.

References

1. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: SMASH: one-shot model architecture search through hypernetworks. In: ICLR 2018
2. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. on PAMI
3. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
4. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV. pp. 801–818 (2018)
5. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: IEEE CVPR (2016)
6. Fernando, C., Banarse, D., Reynolds, M., Besse, F., Pfau, D., Jaderberg, M., Lanctot, M., Wierstra, D.: Convolution by evolution: Differentiable pattern producing networks. In: Proceedings of the GECC 2016. pp. 109–116 (2016)
7. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: ICLR 2019
8. Gong, C., Jiang, Z., Wang, D., Lin, Y., Liu, Q., Pan, D.Z.: Mixed precision neural architecture search for energy efficient deep learning. In: IEEE/ACM ICCAD (2019)
9. Ha, D., Dai, A.M., Le, Q.V.: Hypernetworks. In: In Proceedings of ICLR 2017
10. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. NeuRIPS (2015)
11. Hassner, T., Liu, C.: Dense Image Correspondences for Computer Vision. Springer (2016)
12. Hassner, T., Mayzels, V., Zelnik-Manor, L.: On sifts and their scales. In: IEEE CVPR (2012)
13. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans. on PAMI **37**(9), 1904–1916 (2015)
14. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: IEEE ICCV. pp. 1389–1397 (2017)
15. Ho, J., Kalchbrenner, N., Weissenborn, D., Salimans, T.: Axial attention in multidimensional transformers. arXiv preprint arXiv:1912.12180 (2019)
16. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. NeuRIPS **28** (2015)
17. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. NeuRIPS **29** (2016)
18. Koutnik, J., Gomez, F., Schmidhuber, J.: Evolving neural networks in compressed weight space. In: Genetic and evolutionary computation. pp. 619–626 (2010)
19. Li, Y., Gu, S., Zhang, K., Van Gool, L., Timofte, R.: Dhp: Differentiable meta pruning via hypernetworks. In: ECCV (2020)

20. Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. arXiv preprint arXiv:1901.02985 (2019)
21. Liu, X., Qi, J., Zhang, W., Bao, Z., Wang, K., Li, N.: Recognition method of maize crop rows at the seedling stage based on ms-erfnet model. Computers and Electronics in Agriculture (2023)
22. Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K.T., Sun, J.: Metapruning: Meta learning for automatic neural network channel pruning. In: IEEE ICCV (2019)
23. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE CVPR (2015)
24. Luo, J.H., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. In: IEEE ICCV. pp. 5058–5066 (2017)
25. McGlinchy, J., Johnson, B., Muller, B., Joseph, M., Diaz, J.: Application of unet fully convolutional neural network to impervious surface segmentation in urban environment from high resolution satellite imagery. In: IGARSS 2019
26. Mishra, A., Kumar, S., Kalluri, T., Varma, G., Subramaian, A., Chandraker, M., Jawahar, C.: Semantic segmentation datasets for resource constrained training. In: 7th NCVPRIPG. pp. 450–459. Springer (2020)
27. Molchanov, D., Ashukha, A., Vetrov, D.: Variational dropout sparsifies deep neural networks. In: ICML. pp. 2498–2507. PMLR (2017)
28. Neuhold, G., Ollmann, T., Rota Bulo, S., Kontschieder, P.: The mapillary vistas dataset for semantic understanding of street scenes. In: IEEE ICCV (2017)
29. Pan, Z., Liang, Y., Zhang, J., Yi, X., Yu, Y., Zheng, Y.: Hyperst-net: Hypernetworks for spatio-temporal forecasting. arXiv preprint arXiv:1809.10889 (2018)
30. Romera, E., Álvarez, J.M., Bergasa, L.M., Arroyo, R.: Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. IEEE T-ITS
31. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: IEEE MICCAI. pp. 234–241. Springer (2015)
32. Shaw, A., Hunter, D., Landola, F., Sidhu, S.: Squeezenas: Fast neural architecture search for faster semantic segmentation. In: Proceedings of the IEEE ICCVW 2019
33. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Journal of Artificial Life
34. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: IEEE CVPR (2019)
35. Varma, G., Subramanian, A., Namboodiri, A., Chandraker, M., Jawahar, C.: Idd: A dataset for exproceedings of the eccvploring problems of autonomous navigation in unconstrained environments. In: 2019 IEEE WACV (2019)
36. Yang, C., Yang, Z., Khattak, A.M., Yang, L., Zhang, W., Gao, W., Wang, M.: Structured pruning of convolutional neural networks via l1 regularization. IEEE Access **7**, 106385–106394 (2019)
37. Yu, F., Koltun, V., Funkhouser, T.: Dilated residual networks. In: Proceedings of the IEEE CVPR. pp. 472–480 (2017)
38. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: IEEE CVPR. pp. 2881–2890 (2017)
39. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: IEEE ICCV. pp. 1529–1537 (2015)
40. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)