To Prune or Not to Prune, It is a question: Efficiency in Semantic Segmentation

Zihan Shao
School of Arts and Science
New York University Shanghai
Shanghai, China
Email: zs1542@nyu.edu

Qianyu Zhu
School of Arts and Science
New York University Shanghai
Shanghai, China
Email: qz1086@nyu.edu

Muyang Xu
School of Computer Science and Engineering
New York University Shanghai
Shanghai, China
Email: mx648@nyu.edu

***Abstract*—Our project addresses the pruning method applied to the image semantic segmentation task. Pruning is a technique in deep learning that aids in the development of smaller and more efficient neural networks. We aim to prune the encoder and decoder, respectively, based on the overall symmetric encoder-decoder structure of models for semantic segmentation. This project first uses un-pruned UNet to pre-train on the dataset. We investigate the encoder and decoder's sensitivity in responding to the pruning process with this well-trained model. Based on our analysis, we do grid search on encoder and decoder with different pruning techniques to find optimal parameters for pruning. Our experiments yield a more profound observation on the impact of the general pruning method on semantic segmentation. The full implementation (based on PyTorch) and the trained networks are available at the repository; please refer to the Github link for details.**

`https://github.com/EddyShao/CV-FinalProject`

I. Introduction

Image semantic segmentation has significant application potential in the field of automatic driving. Suppose vehicles can understand the images or videos collected by the camera on the road by accurate and efficient semantic segmentation. In that case, it can provide crucial guidance for obstacle avoidance and path planning, which provides a relatively low-cost information supplement for automatic driving.

In the past decade, most related progress has been achieved by deep neural networks fed with a massive amount of image or video data. For now, methods tackling this task are mainly based on VGG-net and ResNet, with various attention blocks inserted. They achieved high accuracy by capturing enormous details of the object. Still, the exponentially exploding requirement for computation and memory is inefficient, hindering their generalization ability in resource-constrained environments, such as mobile devices or real-time implementation. Our project goal is to reduce the complexity of the model while keeping the loss of accuracy to a low extent.

The increasing contradiction between limited computation power and passion for deep networks boosts the idea of neural network pruning—the task of reducing the size of a network by removing parameters. Pruning has been proved to be an efficient way to enhance efficiency by eliminating redundant connections.

II. Data Set

A. Basic information

This project uses the **Semantic Drone Dataset** from the *Institute of Computer Graphics and Vision* [2], which consists of 400 images from nadir (bird's eye) view acquired at an altitude of 5 to 30 meters above the ground. To carry on semantic segmentation tasks, we use pixel-accurate annotation of 24 classes (1 unlabeled class) to train and test models(see Fig. 1).

Attributes	Value
class	24
number of images	400
resolution (before processing)	6000*4000
resolution (after processing)	768*512
mean (after normalization)	[0.485, 0.456, 0.406]
std (after normalization)	[0.229, 0.224, 0.225]

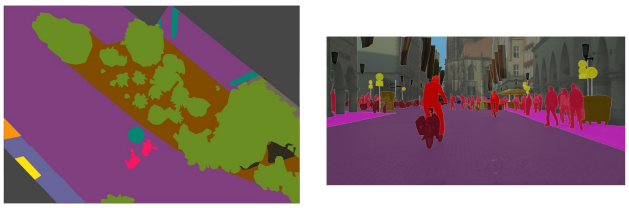
B. Pre-processing

In the pre-processing stage, we transformed the original RGB label masks into gray-scale masks. Each pixel contains a value that represents its corresponding class index. To better hold the training process and let the data fit our model, we resized the original images and the gray-scale masks into lower resolution (768×512 pixels). Furthermore, we normalized the data by setting the RGB mean value as $[0.485, 0.456, 0.406]$ and the standard deviation as $[0.229, 0.224, 0.225]$, which is calculated based on millions of images from the ImageNet datase[13].

C. Drone dataset vs. Cityscapes

Compared with Cityscapes–the traditional semantic segmentation dataset, the drone dataset has a smaller size, providing more flexibility for the training model while significantly reducing computation time complexity [Fig. 1].

Moreover, we can observe that: most label images generated from the bird's-eye view in the drone dataset have distinguishable features. Those features are easily recognized but less easily overlap with each other. Based on the drone dataset's simple feature structure, we speculate that if we cut fewer essential connections in a robust training neural network, whether we can improve the training efficiency and maintain the prediction accuracy at the same time.



Drone Datset Cityscpaes

Fig. 1. Samples from Drone Dataset and Cityscapes

III. Methods

A. Pruning in a Nutshell, with related works

The usual pipeline of pruning is:

1) Train the model with full size
2) decide which weights/filters/channels to prune, and the appropriate sparsity of pruning.
3) Re-train the pruned model to compensate the loss caused by pruning

The general logic of choosing candidates for pruning is salience-based: designing a proper scoring function, issuing every connection a score reflecting its degree of importance to the final prediction, and cutting the paths whose scores are under the self-chosen threshold. The idea of pruning has been investigated since the last 80s, and existing trails differ in sparsity structure, scoring, scheduling, and fine-tuning choices.

1) Sparsity structure

   There are two branches regarding sparsity structure: unstructured pruning and structured pruning. The former will replace the parameters with a small value with zero kernels and yield a sparse model. Theoretically, this approach can help reduce memory and running time to some degree. However, unless the underlying hardware and computing libraries can effectively sustain the unstructured pruning process, it isn't easy to achieve substantial performance improvement after pruning.

   According to "Rethinking the Value of Network Pruning," the sparsity schedule can be classified as "predefined" or "automatic."[3] The predefined sparsity schedule determines that each layer will be compressed using a fixed ratio. While under the "automatic" sparsity schedule, the ratio will be computed referring to the global distribution of parameters; thus, the reduced model after pruning is more unpredictable.
2) Scoring

   The most straightforward scoring function in implementation is cutting redundant connections based on magnitude, described in the paper "Comparing Biases for Minimal Network Construction with Back-Propagation"[4]. However, the 'smaller-norm-less-important' principle is not 100

   Other scoring functions include measuring the second derivative of the loss function relative to the weight of the connection (namely, the Hessian matrix for the weight vector). Then the non-significant weight is trimmed [6][7].

   Since computations of Hessian matrices or their approximations are time-consuming in these methods, several popular indicators are the scaling factor in Batch Normalization or activation layers[8][9]. In general, activation functions like ReLU tend to generate sparse activation; Weights are less likely to be light (currently, as mentioned above, we can make them spare by regularizer).
3) Scheduling

   Different pruning methods substantially contribute to the different pruning amount of the model network. Those can be roughly encapsulated into two types: pruning all desired weight at once and pruning a fixed fraction through several iterative steps[10].
4) fine-tuning

   While fine-tuning is natural, its implementation ways vary a lot. It is most common to iterate the process of training-pruning. Alternative methods include decreasing the pruning ratio (automated gradual pruning)[11].

On the one hand, we would improve the efficiency due to reduced connections, especially the back-propagation computation part. On the other hand, we could drop the model accuracy due to the simplified structure of the model, so it is trained further (known as fine-tuning) to recover. Therefore, it is easy to see, the critical mission of pruning is to find the correct layer to prune $i.e.$, the proper "redundant connections."

Theoretically, it can be fairly easy to judge whether a connection is essential or not. One needs to iterate through all the connections to see how the model's performance is influenced once we eliminate it. However, this method is practically impossible as the complexity to complete the task is $O(2^n)$.

Given this situation, here we have two ways to go:

1) Use certain metrics to issue every layer/node a score and decide which connections to keep/drop. In most cases, this method is less structured and constrained because we treat different node in the same way if they belong to the same type of layers (**e.g.** convolution layer, fully connected layer), this may sometimes lead to erroneous deleting of comparatively more minor but vital parameters in deeper layers;
2) Task-Specific Pruning methods, namely, the pruning method, are designed based on the task and the data set.

In our project, we mainly focus on the second way.

B. Pruning for semantic segmentation

Pruning techniques have been combined closely with features of semantic segmentation. For instance, the performance of semantic segmentation crucially depends on contextual information. Therefore, Wei He proposed a called *context-aware pruning* [12].

Contextual dependency is not the only particularity of semantic segmentation. Usually, models for semantic segmentation are in the "symmetric" encoder-decoder structure. Given this standard structure, it is natural to investigate pruning on them respectively.

1) Sensitivity Testing: First, we are curious about how encoder and decoder act to pruning. Theoretically speaking, the loss graph with respect to sparsity should obey a relation similar to the $sigmoid$ function (Like population model, no rapid growth when sparsity is low and high, but it changes rapidly around $sparsity = 0.5$). However, it may not be the story. We should not dismiss the concatenation part of the encoder-decoder structure. Also, the different results of the encoder and decoder will signal the traits of these models.

2) Grid Search: By doing the grid search, we can see the optimal combination of sparsity for encoder and decoder, respectively. In this part, various kinds of pruning methods are tried.

C. Algorithm design

The current experiences vary primarily in their choices regarding sparsity structure, scoring, scheduling, and fine-tuning. Based on a rough literature review, we applied pruning techniques.

IV. Experiments and Results

A. Baseline

We use **UNet** on Drone dataset (without pruned) [14]. Training information and outcome is as below.

Attrbiutes	Value
Optimizer	Adam
Learning Rate	0.001
Weight Decay	0.0001
Epochs	30
val Loss	2.16
val Accuracy	66.7%
val mIou	0.176

B. Sensitivity of Encoder and Decoder

In our experiments, we applied random pruning on `CONV` layer in Encoder/Decoder respectively with sparsity of from 0 to 1,step $= .05$ and compute the loss of the pruned model on validation set.

Note that:

- Random pruning is not a wise choice in terms of practical usage, but it maximizes the effects of pruning, which is needed in this experiment.
- The bottleneck of UNet is considerably light thus ignored
- The last *conv* layer of the model(**i.e.** output layer) is not pruned.

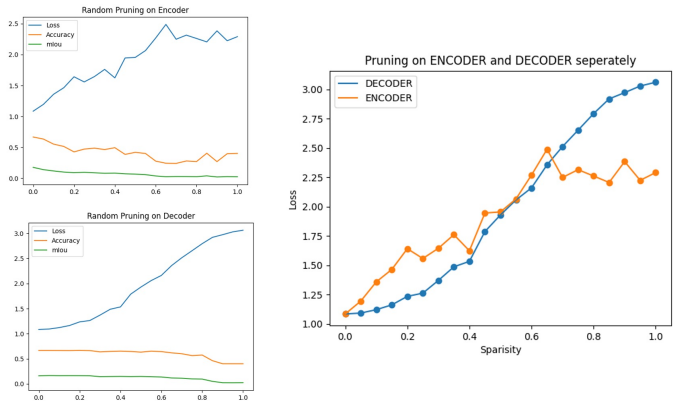


Fig. 2. Results of pruning encoder and decoder seperately

The result of the experiments is shown in Fig. 2. Few conclusions can be drawn from the plots.

1) Generally speaking, the encoder is more sensitive when the sparsity is small (before $sparsity \leq 0.5$), and the conclusion reverses when sparsity grows further.
2) (Pruning on Decoder) The degrading of performance can be roughly fit into a $sigmoid$ function.
3) (Pruning on Encoder) Performance of the model oscillates when the sparsity of the encoder grows. (We have tried many times and got the same result each time). Therefore, it shows that we may reduce the efficiency with no degrading performance by adding more sparsity to the encoder.

Some of the facts are explainable. For 1) it is evident that the decoder's performance is influenced to a more considerable extent than the same sparsity for the encoder. As the decoder is the output part of the model, removing the weights in the decoder disables the ability to predict. (Imagine we put everything, including the bias to be 0, then the only output of the model is 0)

When the sparsity is quite small, removing a small part of the weights will not justify a great difference in the performance. This result is also consistent with the **Lottery Tickets Theory**[15], which indicates no great degrading will occur after dropping 20% of the weights. Also, the performance will not change dramatically when there are nearly no parameters for predicting. Significant change is likely to occur around $sparsity = .5$.

However, the plot generated by pruning purely on the encoder is somehow unexpected. The frequent oscillation seems to contradict one's common sense. Also, the plot is relatively stable in some regions of sparsity (**e.g.** when sparsity is around 0.3).

C. Feature map and Kernel visualization

The different pruning behavior on encoder and decoder perhaps resulted from their only asymmetry, which is the concatenation. Therefore, we plotted out the feature maps we

got after the first down-sampling block **i.e.** the first feature map that will be concatenated to the output(see Fig. 3).

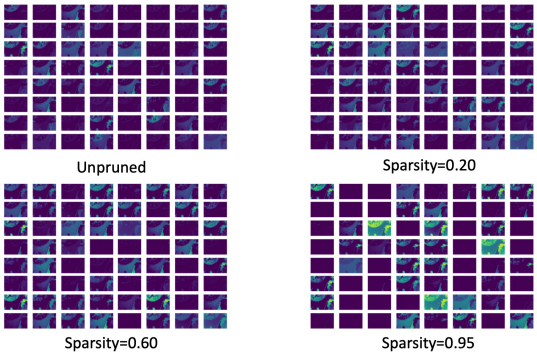


Fig. 3. output after first conv layer

The plots revealed that features are still well extracted despite pruning 20% of the weights. That is possible because we don't have too many parameters at the first two layers, so 20% does not justify a big difference. Though pruned heavily in future blocks, a considerable amount of information is concatenated with up-sampling blocks for final prediction. Therefore, some randomness and oscillations are introduced to the performance.

We have also visualized kernel (see Fig. 4). We can see the kernel changes steeply from the graph when sparsity is set above 50%. However, graphs of the kernel don't provide us with a valid explanation for oscillations.

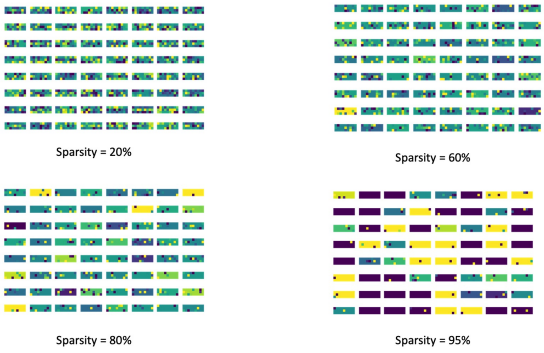


Fig. 4. Kernel of the first double-conv layer

The result of this experiment gives us several implications for our final model.

1) By pruning $\leq$ 30% parameters of the decoding part wisely (**e.g.** apply L_p metric to decide weights/filter/channel prune), we may improve the efficiency without any loss of performance.
2) When pruning the encoder, one should take more trials since there might be oscillations.
3) Structured pruning behaves better than unstructured pruning in our model, partly due to the uneven distribution of weights in deeper convolution layers.

Trial Goal	Sparsity Grid Step	Encoder/ Decoder	Note
Unstructured Random/L1	$10\% \times 10\%$	Both, Grid Search	Global pruning
Structured L1/L2	$10\% \times 10\%$	Both, Grid Search	layer-wise pruning
Pruning Position	20%	Encoder Only	Encoder: Structured L1 Decoder: Unstructured Random

4) We may try not to prune the first two down-sampling layers at all to make sure helpful information is passed to up-sampling layers for prediction.

D. Grid Search

We have the following trials.
Here are the results visualization (see Fig. 5)

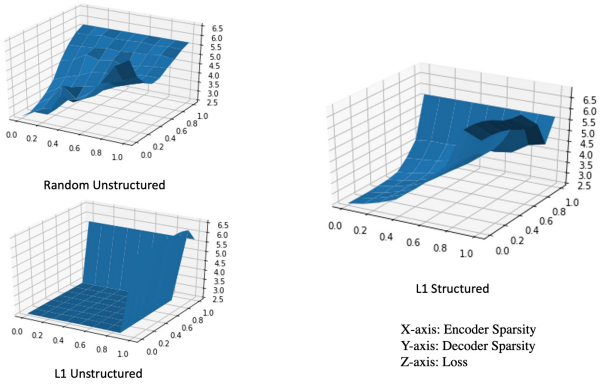


Fig. 5. Results of grid search

The $3D$-Plot for `Random` pruning restate the conclusion we obtained before. The salient oscillations w.r.t sparsity on the encoder is consistent with our decision early. Also, we can see there's no clear correlation between encoder and decoder. The 3D plot we got here is more or less "spanned" by the two curves we obtained before.
The plot of for L_1 unstructured pruning is, however, considered interesting. The plot shows that despite great sparsity, the model's performance doesn't change a lot. The possible reasons are as follows:

- Though the name is fancy, L_1 unstructured pruning is simply comparing the absolute value of every single weight and dropping the one with a small magnitude.
- `UNet` might be too robust for the drone dataset, which results in a tremendous amount of small weights (We manually checked the consequences). Deeper layers are equipped with more parameters, but less of them are vital for the final prediction.
- Those nuisances are dropped by pruning without harming the performance.

Though the outcome of L_1 unstructured pruning is tempting, it would not be practical to prune out 90% of the weights. We have also tried grid search on L_2L_1 structured pruning (the plot of the results are similar, L_1 shown in the fig). Here we can see, the surface generated is much more smooth than random pruning. Applying the L_1/L_2 metric shows that we are indeed making a wiser choice of pruning.

E. Model Proposed

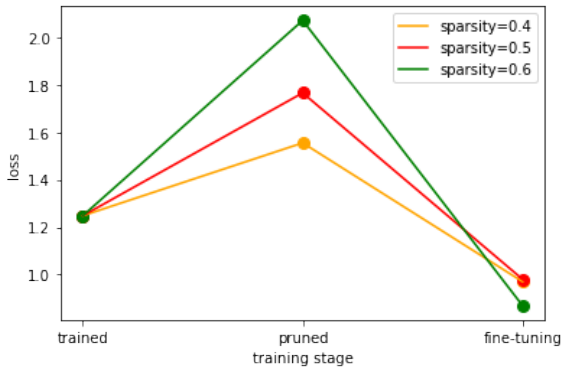


Fig. 6. Loss in different training stage

We choose the best-behaved model settings in grid-search, and fine-tuning by 30 epochs to recover its accuracy loss caused by tuning. We freeze the first half convolution layers in encoder and cut 40%-60% of parameters in the rest convolution layers by L1 pruning, and the results in different training stages are plotted. The result shows that the lossits accuracy can be fully recovered after tuning stage. It is in line with our expectations that loss increases after pruning, and decreases after fine-tuning. But the data also imply that more simplified model may show greater potential during the fine-tuning stage. The model with highest sparsity (=0.6) has the lowest loss after tuning. Compared with the baseline (loss = 2.16, accuracy = 66.7%, mIou = .176), the best result is (.87, accuracy = 72.7%, mIou = .234). Further investigation could be done in this direction.

V. Discussion

A. Reflections and Observations

The encoder shows a more apparent instant response regarding the **small-scale pruning**, while the decoder presents a comparatively delayed reaction. Conversely, the encoder is much less sensitive regarding the **large-scale pruning**, while the decoder's power decays exponentially when the pruning ratio is more significant than 50%.

In terms of UNet structure, the lower layer acts more sensitively toward the pruning process. The entire neural network structure is tolerant to the unstructured L1 pruning method.

B. Future Works

Recalling on all previous progress, we have several anticipations of our future works:

First, we attempt to probe into the connections between image segmentation and video segmentation. Since a video can be regarded as an aggregation of great consecutive pictures, the synergistic effect of those pictures' segmentation may contribute to a video segmentation correspondingly as a whole. It is also essential to figure out a more efficient algorithm for the training process, thus guaranteeing our method is cost-effective and less time-consuming. Therefore, we may want to explore more details: a) whether reducing model usage after pruning can satisfy substantial video datasets while controlling the accuracy loss b) whether the pruning method is effective to all kinds of training images or more determinant to some image features.

Additionally, through the asymmetrical structure of UNet, we probed into the sensitivity of pruning w.r.t different layers at the encoding or decoding section. The "well-organized" neural networks motivated us to intuitively form more explicit controlled trials. Derived from experiment results, for the next step, we intend to figure out how we can apply those intuitive experiment results to a more complex training model which has not as transparent characteristics as UNet.

Acknowledgment

We acknowledge Prof. Robert Fergus for conducting such a miraculously rewarding Computer Vision course this semester, without which we would not be able to complete the above paper.

References

[1] Anwar, Sajid, et. al, *Structured Pruning of Deep Convolutional Neural Networks*, arXiv:1512.08571
[2] Semantic Drone Dataset http://dronedataset.icg.tugraz.at
[3] Liu, Zhang, et. al, *Rethinking the Value of Network Pruning*, arXiv:1810.05270
[4] Li, Hao, et. al, *Pruning Filters for Efficient ConvNets*, arXiv:1608.08710
[5] Ye, Jianbo, et. al, *Rethinking the Smaller-Norm-Less-Informative Assumption in Channel Pruning of Convolution Layers*, arXiv:1802.00124
[6] LeCun, Yann, et. al, *Optimal Brain Damage*, in "Advances in neural information processing systems", pp. 598–605, 1990
[7] Hassibi, Babak and Stork, David G, *Second Order Derivatives for Network Pruning: Optimal Brain Surgeon*, Morgan Kaufmann, 1993
[8] Liu, Zhuang *Learning Efficient Convolutional Networks through Network Slimming*, arXiv:1708.06519
[9] Hu, Hengyuan, et. al, *Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures*, arXiv:1607.03250
[10] Blalock, Davis, et. al, *What is the State of Neural Network Pruning?*, arXiv:2003.03033
[11] Zhu, Michael and Gupta, Suyog, *To prune, or not to prune: exploring the efficacy of pruning for model compression*, arXiv:1710.01878
[12] He, Wei, et. al, *CAP: Context-Aware Pruning for Semantic Segmentation*, in "2021 IEEE Winter Conference on Applications of Computer Vision (WACV)", pp. 959-968, 2021, doi:10.1109/WACV48630.2021.00100
[13] ImageNet, https://image-net.org/index.php
[14] Ronneberger, Olaf, et. al, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, arXiv:1505.04597
[15] Frankle, Jonathan et. al, *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks.*, arxiv.1803.03635