Paper Review
SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY

Muhammad Umar Salman
21010241@mbzuai.ac.ae

Samar Fares
21010173@mbzuai.ac.ae

Yu Kang Wong
21010186@mbzuai.ac.ae

1 Weakness 1

1.1 Summary

In this paper the authors mention that the initialization must be done carefully. Firstly, because if the weights become too large it would cause the gradients to become uninformative. Secondly, because for pruning structurally unimportant connections the saliency measure must be robust to architecture variations. Both these crucial conditions are ensured by using variance scaling methods to initialize the weights. This shows how important the initialization of weights is for the proposed method, however, the authors who throughout the paper have comprehensively gone in so much detail to explain all the intricate math of the proposed method have for the initializations for weights just referenced it to Glorot et al. [1] without giving any explanation of its insights to the reader. Here we will discuss the initialization method that the authors left out to give the readers a more clear understanding of the underlying method.

1.2 Suggestion for Improvement

In Glorot et al. [1] we find out that back-propagated gradients decrease as they move from the output layer towards the input layer right after initialization. Not only that but also that the variance of the gradients decrease as we go backwards in the network. Generally we initialize the weights using the commonly used heuristic:

$$W_{ij} \sim U\left[\frac{-1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$$

where $U[-a, a]$ is the uniform distribution in the interval (-a, a) and n is the size of the previous layer. This standard initialization will cause the back-propagated gradients to be dependent on the size of the previous layer thus causing it to decrease. Therefore, the normalization factor is important while initializing deep neural networks because of the multiplicative effect that goes throughout the layers. So to satisfy our objectives of maintaining back-propagated gradients and back-propagated gradient variances we use the **normalized initialization** proposed in Glorot et al. [1] which is as follows:

$$W_{ij} \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

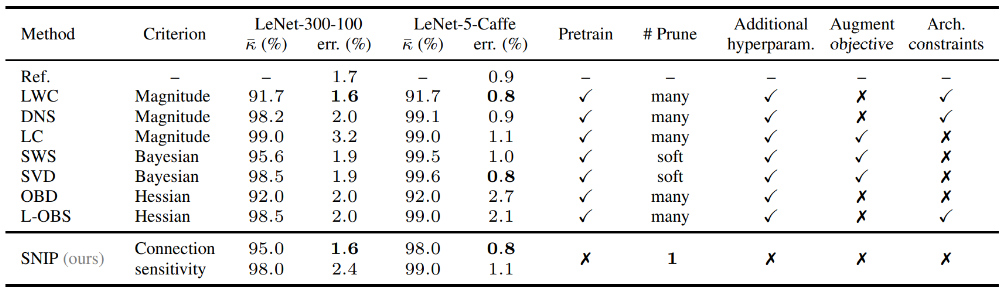
where n_j and n_{j+1} are the size of the input and output layers respectively. Through this variance scaling method of initialization our gradients are no longer dependent on the size of the layers and we manage to achieve both informative gradients as well as our saliency measure being robust to architectural variation.

2 Weakness 2

2.1 Summary

Even though most tables and figures produced by the authors were very detailed and well-constructed, there was a table which was missing some key elements to it, rendering the table unclear and complex. Firstly, the abbreviated methods e.g.(LWC, OBD etc.) should be referenced in the table for more clarity and for the benefit of the reader. Secondly, in the table the caption is mixed along with the analysis of the table rather than the detail of the columns. The analysis of the experiment results should be done in a paragraph which references the table rather than being included in the footnote of the table . Thirdly, the hierarchy levels in the column names make it confusing for readers for e.g. it should be clear that the $\tilde{\kappa}$ and err percentages are being told for the two different LeNet network structures. Lastly, column names are ambiguous causing clarity issues for the reader e.g., “Arch.” and “$\tilde{\kappa}$” which should be in full form or explained in the table footnote. These issues could impact the readers by making them unclear, confused and making them put in more effort to go through the tables.

Below is the image of the table created by the authors.



Method	Criterion	LeNet-300-100		LeNet-5-Caffe		Pretrain	# Prune	Additional hyperparam.	Augment objective	Arch. constraints
		$\tilde{\kappa}$ (%)	err. (%)	$\tilde{\kappa}$ (%)	err. (%)					
Ref.	–	–	1.7	–	0.9	–	–	–	–	–
LWC	Magnitude	91.7	**1.6**	91.7	**0.8**	✓	many	✓	✗	✓
DNS	Magnitude	98.2	2.0	99.1	0.9	✓	many	✓	✗	✓
LC	Magnitude	99.0	3.2	99.0	1.1	✓	many	✓	✓	✗
SWS	Bayesian	95.6	1.9	99.5	1.0	✓	soft	✓	✓	✗
SVD	Bayesian	98.5	1.9	99.6	**0.8**	✓	soft	✓	✓	✗
OBD	Hessian	92.0	2.0	92.0	2.7	✓	many	✓	✗	✗
L-OBS	Hessian	98.5	2.0	99.0	2.1	✓	many	✓	✗	✓
SNIP (ours)	Connection sensitivity	95.0	**1.6**	98.0	**0.8**	✗	**1**	✗	✗	✗
		98.0	2.4	99.0	1.1					

Table 1: Pruning results on LeNets and comparisons to other approaches. Here, “many” refers to an arbitrary number often in the order of total learning steps, and “soft” refers to soft pruning in Bayesian based methods. Our approach is capable of pruning up to 98% for LeNet-300-100 and 99% for LeNet-5-Caffe with marginal increases in error from the reference network. Notably, our approach is considerably simpler than other approaches, with no requirements such as pretraining, additional hyperparameters, augmented training objective or architecture dependent constraints.

Figure 1: Image of table by authors.

2.2 Suggestion for Improvement

Below we have reconstructed the table as an example of more structure and clarity.

Reference	Method	Criterion	LeNet-300-100		LeNet-5-Caffe		Pretrain	# Prunes	Additional	Augment	Architecture
			$\tilde{\kappa}$(%)	err.(%)	$\tilde{\kappa}$(%)	err.(%)			Hyperparams	Objective	Constraints
Han et al. [2]	LWC	Magnitude	91.7	1.6	91.7	0.8	✓	many	✓	✗	✓
Guo et al.[3]	DNS	Magnitude	98.2	2.0	99.1	0.9	✓	many	✓	✗	✓
Idelbayev et al.[4]	LC	Magnitude	99.0	3.2	99.0	1.1	✓	many	✓	✓	✗
Ullrich et al.[5]	SWS	Bayesian	95.6	1.9	99.5	1.0	✓	soft	✓	✓	✗
Molchanov et al.[6]	SVD	Bayesian	98.5	1.9	99.6	0.8	✓	soft	✓	✓	✗
LeCun et al.[7]	OBD	Hessian	92.0	2.0	92.0	2.7	✓	many	✓	✗	✗
Dong et al.[8]	L-OBS	Hessian	98.5	2.0	99.0	2.1	✓	many	✓	✗	✓
	SNIP	Connection	95.0	1.6	98.0	0.8	✗	one	✗	✗	✗
	(Ours)	Sensitivity	98.0	2.4	99.0	1.1					

Table 1: **Comparing pruning results on LeNets to other approaches.**Where $\tilde{\kappa}$ is the sparsity level percentage. Here "many" means an large number of prunes and "soft" refers to soft running in Bayesian methods. Column Pretrain is a boolean whether the method requires pretraining or not. Column Architectural Constraints refers to whether the method is robust to architectural variation and column Augment Objective is if the methods augment the training objective to handle pruning or not. Also green values represent the smallest error percentages among all methods on both LeNet network structures.

3 Weakness 3

3.1 Summary

Another weakness of the paper was right after the “Experiments” section the authors go directly to the “Discussion and Future Work” section without giving any conclusive remarks or a “Conclusion” section. The authors here just assume that the reader after reading the “Experiments” section were able to infer what the authors wanted to conclude and justify from their paper. This however, creates an incomplete feeling and impacts negatively as the reader wants to read the final comments of the authors on the paper before the authors go on to discuss where their ideas can fill gaps and open up new possibilities in the future. These ending conclusive remarks not only give a holistic view to a complete research paper but also ensures the reader on the justification and success of what the authors are trying to present. Here we will write the “Conclusion” section which the authors have missed.

3.2 Suggestion for Improvement

In this work, we presented a new pruning approach which prunes a network at initialization prior to training. From the experiment results we can see that on the LeNet network structures our proposed method after pruning extreme sparsity levels compares better to various pruning algorithms. Not only that but it does better than most methods without any complex pretraining which isn’t the case for the other methods. It also has a single-shot pruning step which is done at initialization whereas all other methods require a large number of expensive pruning cycles. This method also passes the test of robustness to variation in architecture where it is tested on multiple architectures and still manages to get no loss in performance even after reducing a significant number of parameters. Thus we can see that our proposed method is not only simple and versatile but also is efficient and effective and we believe this may open up new possibilities in and beyond pruning.

4 Weakness 4

4.1 Summary

The authors conducted their experiments on relatively small datasets and models with small number of learnable parameters which don’t showcase the main motivation of this paper which is to prune large neural networks with minimal or no loss in performance. We can see that the authors have evaluated the proposed method on MNIST, CIFAR-10 and Tiny-ImageNet with different network architectures in the "Experiments" section whose number of images are in the thousands and are not very diverse. Even for different models tested in the paper such as LSTMs and GRUs we can see that the total learnable parameters are in hundred thousands whereas the models built nowadays have parameters in the millions if not billions. Reducing already small number of parameters or images which are small in number and alike have an undermining and uncertain impact on the reader.

4.2 Suggestion for Improvement

To overcome this weakness, we would like to suggest the author use a larger dataset and models with larger learnable parameters so that the difference of number of parameters before and after pruning are significantly large. This will not only justify the fact that the proposed method is effective in terms of performance but will also give confidence to readers and future researchers who may choose to use it in the future. Other than that we would like to recommend the authors to use a larger and more diverse dataset such as ImageNet which contains 14 million images and over more than 20000 categories. A good performance result on such a large and diverse dataset would showcase the confidence and consistency of the proposed method to the readers.

References

[1] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *AISTATS*, 2010.

[2] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *NIPS*, 2015.

[3] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *NIPS*, 2016.

[4] Yerlan Idelbayev and Miguel A. Carreira-Perpi. “Learning-compression” algorithms for neural net pruning. *CVPR*, 2018.

[5] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *ICLR*, 2017

[6] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. *ICML*, 2017a

[7] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. *NIPS*, 1990.

[8] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *NIPS*, 2017.