Proceedings of the 11th International Conference on Applied Informatics
Eger, Hungary, January 29–31, 2020, published at http://ceur-ws.org

Finding Dense Supermasks in Randomly Initialized Neural Networks

Csanád Sándor[a, b]

[a]Faculty of Mathematics and Informatics,
Babeş-Bolyai University, Cluj-Napoca, Romania
csanad.sandor@cs.ubbcluj.ro

[b]Robert Bosch SRL, Cluj-Napoca, Romania

Abstract

Recent works on network pruning [16, 12] showed that randomly weighted neural networks contain supermasks (subnetworks) the performance of which is comparable with similarly sized, trained networks. These results underpin the Lottery Ticket Hypothesis [3]: the effectiveness of deep neural networks rely on lucky initialization.

While these works define sparse supermasks, we demonstrate that dense supermasks can also be found by applying structured pruning. We remove components from randomly weighted neural networks – neurons from fully connected layers – such that the loss of the networks decreases continuously. This results in smaller, dense networks whose accuracy is higher than their initial version.

Keywords: deep learning, network pruning, supermasks, subnetworks

1. Introduction

The trend to improve the accuracy of modern deep neural networks is to increase the number of parameters and the number of layers [7, 15, 5]. This practice, with the help of bigger datasets, more memory and computing power, increased the accuracy of neural networks on challenges such as the ImageNet Large Scale Visual Recognition Challenge [13]. However, increasing the network size demands gigabytes of memory and billions of floating point multiplication during inference that is usually not available in resource limited devices. Moreover, different works pointed out that these neural networks are over-parameterized [2, 1]: some of the parameters are redundant. Eliminating them from the network causes no drop in the accuracy.

Since there is an increasing need to apply neural networks on resource limited devices (autonomous drones, cars, mobile phones or other embedded devices), pruning methods were developed that aim is to eliminate redundant parameters [4, 10, 6, 14, 11]. These techniques estimate the importance of parameters (or groups of parameters) and remove the less important ones such that the loss does not decrease. Network pruning can be summarized in the following steps: (1.) a randomly initialized network is trained on a dataset using some optimization method (e.g. stochastic gradient descent), (2.) importance values are estimated and assigned to the trained weights (e.g. the magnitude of the parameters are considered as importance values), (3.) the less important parameters are eliminated from the network, (4.) the pruned network is retrained to regain its original accuracy. Steps (2.)-(4.) could be repeated multiple times to further decrease the network size.

Pruning results show that smaller networks are able to represent complex distributions but current optimization methods fail to tune their weights adequately. However, optimizing in larger dimensional space and eliminating the parameters afterwards can solve the problem.

Contrary to this, a recent work pointed out that smaller, sparse networks can be trained as good as their dense counterparts, if their weights are initialized properly [3]. The authors argue that optimization methods have more success on large networks only because these chance of having luckily initialized subnetworks is higher. In the paper these subnetworks are called lottery tickets. Training these luckily initialized subnetworks leads to the same accuracy as their dense counterparts.

Following the work of [3], [16] demonstrates that randomly initialized networks already contain subnetworks – called supermasks – with accuracy far better than chance. The authors highlight that the parameters of these subnetworks *are not trained at all.* They freeze the network weights (with the initial values) and find subnetworks with high accuracy. Moreover, [12] presents an edge-popup method that is able to efficiently find subnetworks (in randomly initialized, untrained networks), that accuracy is comparable with their dense, trained counterpart.

Our work is inspired by the results of [16, 12], but instead of examining sparse subnetworks, we demonstrate that randomly initialized, untrained networks contain dense subnetworks with an accuracy far from chance. We experimentally demonstrate that a randomly initialized LeNet-300-100 architecture contains dense subnetworks that accuracy is around 40% on the MNIST dataset. This value is far better then the 10% accuracy of its unpruned counterpart. Moreover, we show that a randomly initialized Wide-LeNet (a 2 layer FCN with increased size) contains dense subnetworks with an accuracy above 60%.

While this work studies fully connected networks on a small dataset we would like to extend these experiments to convolutional neural networks and larger datasets as well.

2. Pruning methods

To prune neurons from a network, we compare the magnitude based pruning with the linear filter ensembles (LFE) method. As a control case, we also apply random pruning where neurons are removed randomly from the network. In the following we briefly present the magnitude and LFE pruning methods.

Magnitude pruning uses the parameter magnitude as importance value: parameters with small magnitude have less importance and their elimination chance is higher. In case of structured pruning, the importance of the i^{th} neuron is estimated by calculating its L2 norm:

$$\theta_i = \left[\sum\nolimits_j \mid w_{i,j} \mid^2\right]^{\frac{1}{2}}$$

where $w_{i,j}$ denotes the j^{th} parameter of the i^{th} neuron and θ_i is the estimated importance.

Linear filter ensembles [14] estimates the importance of different architectural components by considering the network loss. Based on the filter importance values, $p\%$ of the components are removed from the network. While [14] computes the optimal p by evaluating the pruned network on a validation set, we simply set this to a constant value. Therefore, in each pruning iteration $p\%$ of the neurons are removed from the layer.

LFE estimates the importance of the filters (or neurons) by evaluating the network loss using filter ensembles. A filter ensemble is a group of filters (in case of fully connected layers a group of neurons) from a given layer. Each ensemble is represented by a $\boldsymbol{z}_i$ binary vector, that defines which filters should be used during the network evaluation. For each filter ensemble an s_i score is assigned that is calculated from the network loss. Ensembles with small loss get high score while large loss produces smaller scores. Given the $\{\boldsymbol{z}_i, s_i\}$ pairs, $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N]^T$ matrix and $\boldsymbol{s} = [s_1, \ldots, s_N]^T$ vector is constructed and the $\boldsymbol{Z} \cdot \boldsymbol{\theta} = \boldsymbol{s}$ equation is solved by minimizing the Euclidean 2-norm:

$$\mathcal{L} = \|\boldsymbol{s} - \boldsymbol{Z} \cdot \boldsymbol{\theta}\|\,.$$

Finally, $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_N]$ is used as importance indicators, where large θ_i means more importance value.

3. Experiments

We experiment with LFE, magnitude and random based pruning on the LeNet-300-100 [8] architecture and the MNIST dataset [9].

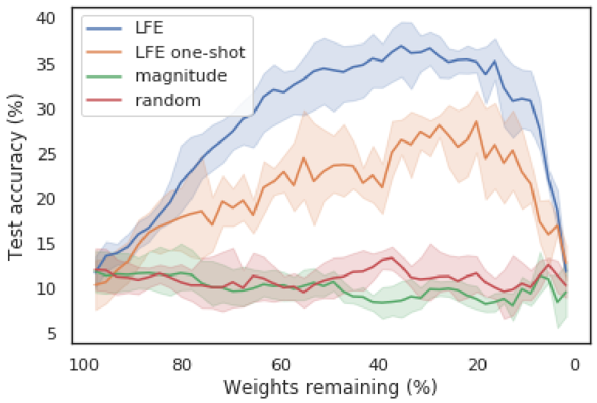

Figure 1: The accuracy of LeNet-300-100, as more and more neurons are removed from the network. The curves present the results of iterative (LFE, magnitude, random) and one-shot pruning (LFE one-shot), using linear filter ensembles, magnitude and random pruning methods.

Experimental setup

MNIST is a dataset that contains grayscale images of handwritten digits ranging from 0 to 9. The dataset contains 60 000 samples in total: 50 000 images belongs to the training set and 10 000 to the test set. Each image has a size of 28×28 pixels and the digits are centered and self-normalized on them.

We experiment with a 3-layer fully connected network, called LeNet-300-100 [8] on the MNIST dataset. This network has an input layer that contains 784 units – one unit for each pixel from the MNIST image. The input layer is followed by two fully connected layers, containing 300 and 100 units respectively and use the ReLU activation function. The output layer contains 10 units and uses softmax activation function. Since we are measuring the performance of randomly initialized, untrained networks, we do not apply any training on them (we do not modify the parameters, only remove them if required). The weights values correspond to the initial values sampled from normal distribution with 0 mean, 0.1 standard deviation. All biases are set to 0.

Using the pruning methods (without training the network), we iteratively remove 6 and 2 units from the first and second layers. Therefore, the network size shrinks from 100% to 1.77%. During each iteration, we measure the network loss and accuracy on the test set. We also experiment with one-shot pruning using LFE: instead of removing the neurons iteratively, the network size is decreased to the target value simultaneously. In each pruning method we repeat the process 5 times and report the average loss and accuracy, as well as the min and max of these values. To apply LFE pruning, we insert mask layers after the hidden layers.

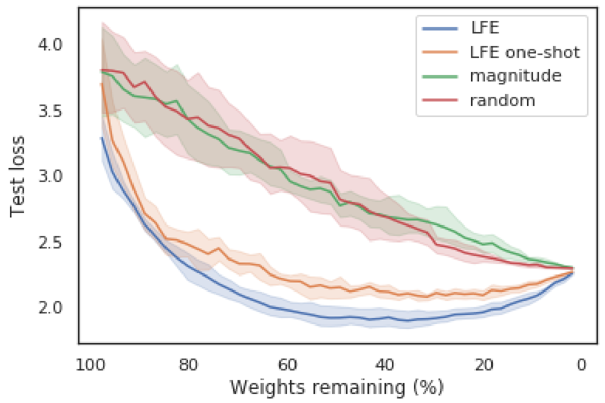

Figure 2: The loss of LeNet-300-100, as more and more neurons are removed from the network. The curves present the results of iterative (LFE, magnitude, random) and one-shot pruning (LFE one-shot), using linear filter ensembles, magnitude and random pruning methods.

These mask layers define the active and inactive units (neurons) from the previous layer: units with 0 masks are turned off, therefore, the weights of that unit could not contribute to the network output. We use the cross entropy loss function to calculate the score of different filter ensembles.

3.1. LeNet-300-100 results

First, we experiment with the LeNet-300-100 fully connected network. Results of the experiment is presented in Figure 1 and Figure 2. As Figure 1 shows, the network accuracy with random pruning remains around the initial 10%. This value is expected in case of a classification problem with 10 categories. This corresponds to the intuition that a randomly initialized network after random pruning became a smaller but still random network.

Surprisingly, the magnitude based pruning could not increase the network accuracy at all. While [16] reports huge accuracy increase with magnitude based, unstructured pruning, this does not apply for magnitude based structured pruning. With magnitude based pruning, the accuracy curve is similar to the curve of random pruning. However, this behavior is expected as well: the parameters are initialized with normal distribution, resulting near equal L2 norm per unit. Near equal importance values are not useful for pruning parameters.

In contrast to this, LFE pruning (iterative and one-shot as well) significantly increases the network accuracy, as more and more units are removed from the network. Applying one-shot pruning, the network accuracy peaks around 29%, where 80% of the parameters are removed from the network. This increase is more

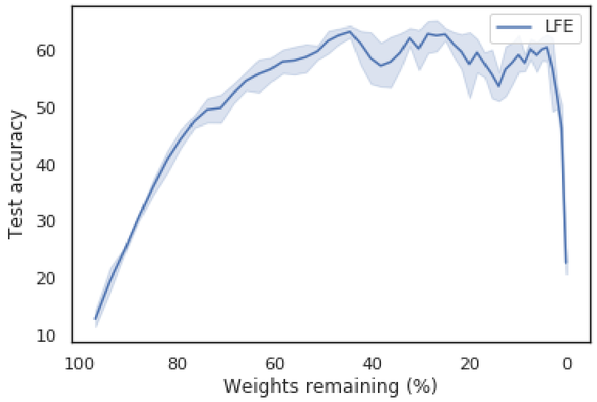

Figure 3: Change of accuracy as more and more parameters are removed from the untrained Wide-LeNet: a 2 layer fully connected network with 3010 and 1010 hidden neurons.

significant in case of the iterative LFE: the network accuracy reaches $37\% \pm 4\%$, where the network size is only 35.34% of the initial network (108 and 36 neurons in the first and second layers respectively). After this value, the accuracy quickly decreases and stops at the initial 11.9% with 1.77% of the initial network size.

Figure 2 shows the loss of the networks during parameter pruning. With LFE pruning (iterative and one-shot as well) the network loss decreases as expected: it reaches a minimum value and starts to increase afterwards. However, random and magnitude pruning produces a continuously decreasing loss which is in contrast with the accuracy curve. One would expect that randomly removing parameters from a random network will not influence the network loss. However, as more and more parameters are eliminated, the network output (probabilities) becomes more and more uniform (the network assigns near 0.1 probability for all the 10 digits). This results decreasing cross entropy loss that asymptotically reaches the value $\mathcal{L} = -\log(0.1) \sim 2.3025$.

3.2. Wide-LeNet results

Motivated by the results of [12], we also experiment with the pruning of (untrained) Wide-LeNet: a 2 layer fully connected network with 3010 and 1010 units in the first and second hidden layers. Since this network has far more parameters, it should have higher chance to contain luckily initialized subnetworks. Based on the success of LFE pruning, we iteratively remove 60 and 20 units and report the results in Figure 3. Pruning 55.44% of the parameters, the network produces $63.47\% \pm 1.56\%$ accuracy on the MNIST test set while at 8.98% network size – 250 and 90 neurons – the network accuracy is still above 60%.

These results validates the hypothesis that randomly initialized networks contain supermasks – with accuracy far from random – not only on sparse form but in dense form as well.

4. Conclusion

In this work we experimentally showed that randomly initialized, untrained neural networks contain dense subnetworks with an accuracy far better than chance. Using magnitude and LFE pruning, we iteratively pruned the randomly initialized LeNet-300-100 architecture and measured the network loss and accuracy. With LFE pruning, we found dense subnetworks with an accuracy of 37% on the MNIST dataset. Moreover, applying the same pruning method on Wide-LeNet, we achieved 63.5% accuracy, without training the parameters at all.

In the future we would like to expand these experiments to bigger networks and real world datasets as well.

References

[1] Dauphin, Y. N., and Bengio, Y. Big neural networks waste capacity. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings* (2013), Y. Bengio and Y. LeCun, Eds.

[2] Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and de Freitas, N. Predicting parameters in deep learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Red Hook, NY, USA, 2013), NIPS'13, Curran Associates Inc., p. 2148–2156.

[3] Frankle, J., and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (2019), OpenReview.net.

[4] Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (2016), Y. Bengio and Y. LeCun, Eds.

[5] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016* (2016), IEEE Computer Society, pp. 770–778.

[6] He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18* (7 2018), International Joint Conferences on Artificial Intelligence Organization, pp. 2234–2240.

[7] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International*

Conference on Neural Information Processing Systems - Volume 1 (Red Hook, NY, USA, 2012), NIPS'12, Curran Associates Inc., p. 1097–1105.

[8] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[9] LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist 2* (2010).

[10] Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *CoRR abs/1608.08710* (2016).

[11] Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018), OpenReview.net.

[12] Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's Hidden in a Randomly Weighted Neural Network? *arXiv e-prints* (Nov. 2019), arXiv:1911.13299.

[13] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252.

[14] Sándor, C., Pável, S., and Csató, L. Pruning CNN's with linear filter ensembles. *arXiv e-prints* (Jan. 2020), arXiv:2001.08142.

[15] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition, 2014. cite arxiv:1409.1556.

[16] Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 3592–3602.