Neural Network Pruning using Quality and Diversity of Neurons

Project-II (EE47008) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology (Hons.)

in

Electrical Engineering

by

Kushagra Chitkara

(19EE10036)

Under the supervision of

Professor Jiaul Paik

Department of Electrical Engineering

Indian Institute of Technology Kharagpur

Spring Semester, 2022-23

April 29, 2023

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 29, 2023 (Kushagra Chitkara)

Place: Kharagpur (19EE10036)

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Neural Network Pruning using Quality and Diversity of Neurons**" submitted by **Kushagra Chitkara** (Roll No. 19EE10036) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology (Hons.) in Electrical Engineering is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2022-23.

Date: April 29, 2023
Place: Kharagpur

Professor Jiaul Paik
Centre for Educational Technology
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Kushagra Chitkara** Roll No: **19EE10036**

Degree for which submitted: **Bachelor of Technology (Hons.)**

Department: **Department of Electrical Engineering**

Thesis title: **Neural Network Pruning using Quality and Diversity of Neurons**

Thesis supervisor: **Professor Jiaul Paik**

Month and year of thesis submission: **April 29, 2023**

As neural networks grow in popularity, more complicated neural networks emerge. Everyone can't have the resources to implement these state-of-the-art networks. To make these models more accessible, we aim to find subnetworks within such trained neural networks to reduce the network's overall size without compromising accuracy. Pruning is an idea that has been around since the 1990s. Recently, researchers have proved that such subnetworks exist by pruning specific weights in the network. However, this approach has limited practical implications because while performing algebraic calculations via GPUs, the matrix being fed would still be roughly of the same size, despite being sparse. We propose to improvise on this idea and drop entire neurons instead. Hence, the matrix would now be condensed for optimisation. To achieve this, we are using a trained neural network, applying DPP (Determinantal Point Process) on each layer, and then comparing accuracies. To further improve upon accuracies, we are retraining our sparse network as the now optimised calculations would make this step less time-consuming.

Acknowledgements

I would like to express my gratitude to my supervisor Professor Jiaul Paik and my mentor Mr. Atif Hassan for their exceptional guidance and support, through all the stages of project. They have motivated me to explore various ideas and provided me multiple papers and resources. They helped me with all my problems faced during the project.

I am grateful to the Centre for Educational Technology at IIT Kharagpur for providing me with this wonderful opportunity to complete this project. Many thanks to everyone who helped make this project a reality.

Contents

List of Figures

Introduction

The development of ever-larger neural networks has been crucial to the advancement of deep learning. However, larger models are more cumbersome to distribute because they take up more space in storage. They're also computationally expensive and require state-of-the-art resources. This is a huge concern for companies looking to build real-world applications out of these models. To save space, neural networks can be pruned to remove excess weights. "Pruning" means eliminating extraneous neurons or weights in machine learning.

A neural network can be pruned in various ways. One option is to prune weights by setting their individual parameters to zero and making the network sparse. This way, we can preserve the same basic structure of the model while reducing the number of parameters. Another option could be to prune entire nodes. By doing so, the network's overall architecture might be simplified while still attempting to replicate the accuracy of the original, more robust system.

Weight-based pruning has gained popularity as it can be performed easily without hurting network performance. Hence, in 2018, researchers from MIT set out to find the extent to which we could prune a network without losing out on accuracy. They started out with a neural network and trained it just short of complete accuracy. After that, they pruned the network parameters till only a pre-defined percentage of the network remained. After this step, they reinitialised the weights of the remaining

neural network to the original weights, then trained this sparse network again. This was followed by another step of pruning and reinitialising. After each iteration, the accuracy of the network was tested, and the process was carried out until there was no significant drop in accuracy. Using this methodology, the authors were able to prune 94% of the neural network parameters without any loss in accuracy.

However, the objective of their research was accuracy-focused and had little practical implications. Since they were not pruning complete neurons, the matrix formulated for algebraic calculations was sparse and not optimised. Hence, we propose a methodology to prune entire nodes without dropping the model's overall accuracy. This would result in a condensed matrix, making it more efficient for calculations and hence of a more practical use. To choose the best neurons, they must satisfy two properties; they should be contributing to the matrix and should be diverse. To contribute to the matrix, it is relatively simple that a neuron should have a high activation value. We use an algorithm called the Determinantal Point Process (DPP) to ensure both of these properties. We then prune the network, check its accuracy, retrain the network, and finally compare its accuracy to the untrained network and the initial accuracy.

The rest of the report is as follows: we provide a brief overview of relevant works. The network and dataset specifications are described in Section 3. In Section 4, we detail our implementation of the DPP algorithm. Our findings are discussed in Section 5. In Section 6, we wrap up the paper and discuss our long-term goals.

Related Work

Neural Network pruning is a technique based on a fairly straightforward concept: are there any redundant parameters in the network that one could remove? Although a specific idea, dozens of papers with increasingly complex algorithms are published each year for identifying said superfluous parameters.

The most basic framework for pruning is to train, prune, and then fine-tune the network. We could play around with the pruning, depending on the structure to prune (like weight parameters, neurons, etc.) and the criteria for pruning (like weight magnitude criteria, gradient magnitude criteria, or local/global pruning). This classic framework, exemplified by (Han et al., 2015) , serves as a basis for several works such as (Li et al., 2016), (Liu et al., 2017), (Molchanov et al., 2016), (Wen et al., 2016).

Some works took inspiration from this classical framework and started to bring in their own modifications. Gale et al. have advanced the notion of iterations by progressively deleting a growing number of weights throughout the training phase, maximising the benefits of iterations, and eliminating the entire process of fine-tuning. He et al. reduced the prunable filters to 0 at each epoch. They did not prevent the weights from learning, so that their weights could grow back after pruning while sparsity is enforced during training. Lastly, Renda et al. use a technique

to completely retrain a network after pruning. This retraining has proven more effective than simple fine-tuning, albeit at a substantially higher price.

Some works have preferred to prune before training in order to speed up the process, avoid fine-tuning, or avoid any modification to architecture during or after training. Inspired by works like LeCun et al. or Mozer and Smolensky], many publications decided to prune at initialisation like (de Jorge et al., 2020), (Wang et al., 2020), and (Hayou et al., 2020).

Another work that emerged from the relationship between pruning and initialisation was the "Lottery Ticket Hypothesis" by Frankle and Carbin. This hypothesis states that a dense neural network with random initialization contains a subnetwork that is initialised so that, when trained in isolation, it can achieve the same test accuracy as the original network after training after no more than the same number of iterations. Multiple works have expanded, stabilised, or studied this hypothesis like (Desai et al., 2019), (Frankle et al., 2019), (Malach et al., 2020), (Morcos et al., 2019), (Zhou et al., 2019).

Over the years, there have been many schools of thought regarding the best way to prune a neural network. We take inspiration from recent advances in neural network pruning in our methodology. Renda et al.] set the basis for retraining a neural network after pruning, but they opted for a simple weight-based criteria for selecting parameters. Our proposal to purely select neurons and consider the diversity of said neurons has not been subject to any specialised study in the domain of neural network pruning.

Model Architecture and Dataset

While we could have chosen virtually any neural network, we took inspiration from (Frankle and Carbin, 2018) and chose an architecture similar to one of their experiments. For the data, we went with Fashion-MNIST or the FMNIST dataset. FMNIST is a dataset much like MNIST, with 60,000 training samples of 28*28 pixel images of various fashion apparel and ten categories mapping these images to multiple categories like Trousers, Coats, or Sandals. The reason for choosing FMNIST over the MNIST dataset was influenced by a number of factors that the creators of FMNIST have illustrated (Xiao et al., 2017).

First, the MNIST dataset has been deemed 'too easy,' and even classic machine learning algorithms could achieve an accuracy greater than 97%. For our use, it is imperative that we employ a dataset that actually tests the network and reports a drop in accuracy in case of any flaws. Secondly, the MNIST dataset has been too overused, with many neural network pruning publications having reported their findings on MNIST. We would like to see if our technique performs well on not-so-common datasets as well. That being said, comparing our accuracy results with similar pruning techniques on the same dataset is a good idea. Hence, in our future work (5.4), we plan to expand this methodology to several other datasets and architectures.

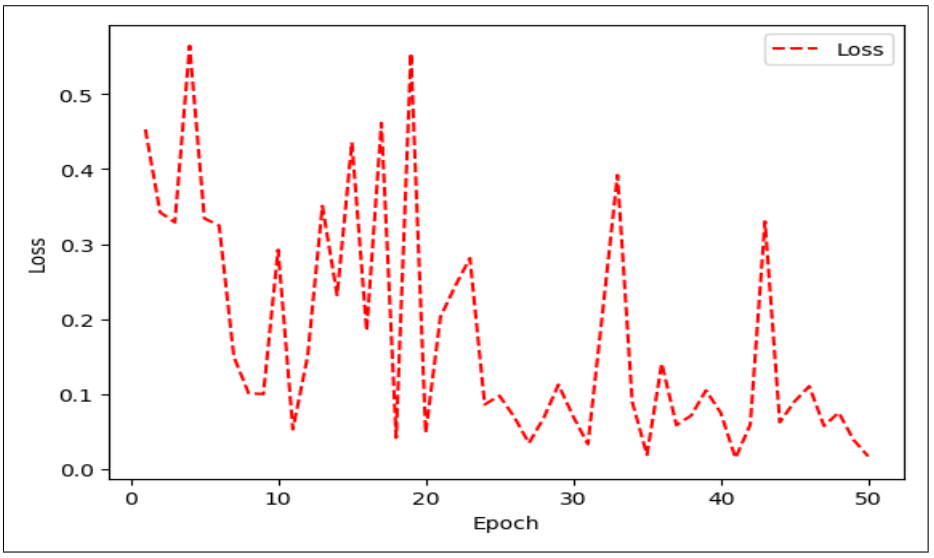

FIGURE 3.1: Negative Log Likelihood Loss for training data after 50 epochs

The architecture of our experiment is a multilayer perceptron, which has 784 neurons in the input layer, representing the flattened 28*28 pixels of our input layer. Two hidden layers follow this input layer, with 300 and 100 neurons, as was the case in (Frankle and Carbin, 2018). Finally, our output layer has ten labels for the ten categories of fashion apparel. We also take a batch size of 32, a ReLU activation, so that we do not face any case of vanishing gradients, as popularised by Nair and Hinton. Further, we use a negative loss likelihood function and Adam optimizer (Kingma and Ba, 2014). We test our hyperparameters and decide to train for 50 epochs at a learning rate of 3×10^{-4}. As we could see in Fig 3.1, the loss function stabilises when reaching the 50^{th} epoch.

Methodology

After an initial training of our model, we wish to prune the network and then retrain it. As discussed in Related Work (2), our technique opts for pruning entire neurons instead of weight parameters. The question still remains: How do we choose which neurons to prune? The neurons chosen should satisfy two essential properties; they should contribute to the network's ultimate output and be diverse. To quantify the contribution to the network is straightforward; we could choose neurons that have higher activation values. Even though the magnitude criterion is simple, it is still used in modern works (Gale et al., 2019), (Han et al., 2015), (Renda et al., 2020), making it an important part of the domain. Implementing diversity amongst neurons is the tricky part; for this, we employ the Determinantal Point Process algorithm (Kulesza, 2012).

4.1 Determinantal Point Process

4.1.1 Functioning

The Determinantal Point Process, or DPP, is an algorithm that was proposed by Kulesza in 2012. In this algorithm, we construct a K-by-K matrix L, where *L[i][j]* is a function of a_i and a_j, Thus:

$$L[i][j] = f(a_i, a_j)$$

Here, a_i and a_j would represent the i^{th} and j^{th} neuron for us. Once we have constructed this matrix, we wish to find a submatrix L_y from the matrix L. For any submatrix L_y, the probability of selecting it would be:

$$P(y) = det(Ly)/det(L + I)$$

where I is an identity matrix. We want to select a submatrix L_y so that *P(y)* is maximised. For our purpose, the size of this new submatrix would represent the percentage of neurons we wish to keep in our model. Let's assume we have a 2-by-2 submatrix, as shown below:

$$Ly = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$$

Since *det(L+I)* would be common to every such submatrix, *P(y)* is proportional to the absolute value of L_y. For the given submatrix:

$$det(L_y) = a_1.a_4 - a_2.a_3$$

Hence, to maximise *det(L_y)*, we either need to have $a_1.a_4$ or $a_2.a_3$ to have a high value; at the same time, we do not wish for the pair of neurons to have similar values. If we assume a_i is proportional to the activation value of the i^{th} neuron, this translates to the fact that neurons with higher activation values are being promoted while simultaneously penalising pairs of neurons with similar values. This way, we could ensure both a high activation value and diversity amongst neurons.

4.1.2 Kernel for DPP

Now that we have looked at how DPP functions, let's look at how we constructed the kernel matrix L. We construct a kernel for each hidden layer, which is 2 in our architecture. Each kernel L would be a matrix of size n-by-n, n representing the number of neurons in that particular hidden layer. The values of this matrix are given by the formula.

$$L[i][j] = q_i.C_ij.q_j$$

Here q_i represents the quality of the i^{th} neuron. The quality of a neuron is nothing but its average activation value over all the training samples. Upon using F-MNIST (Xiao et al., 2017), we get 60,000 training samples. So the quality of the neuron would be the activation value averaged for 60,000 training samples.

C_{ij} represents the correlation between two vectors, i and j, which is simply a vector form of the activation values obtained over all the training samples. Hence, it is the correlation between two vectors of length 60,000 containing the activation values of neurons i and j. We use the Pearson Correlation Coefficient for our research.

4.2 Pruning

To prune the neural networks, we create separate kernels for each hidden layer. We then choose a percentage of neurons to keep. Inspired by the various literature we surveyed like (Renda et al., 2020) and (Frankle and Carbin, 2018), we decided to save 25% of the network intact. Although many publications show that such winning subnetworks may exist for even sparser networks, 25% would be a great starting point to experiment, and then we would try to further prune the networks to about 5% in our Future Work (5.4).

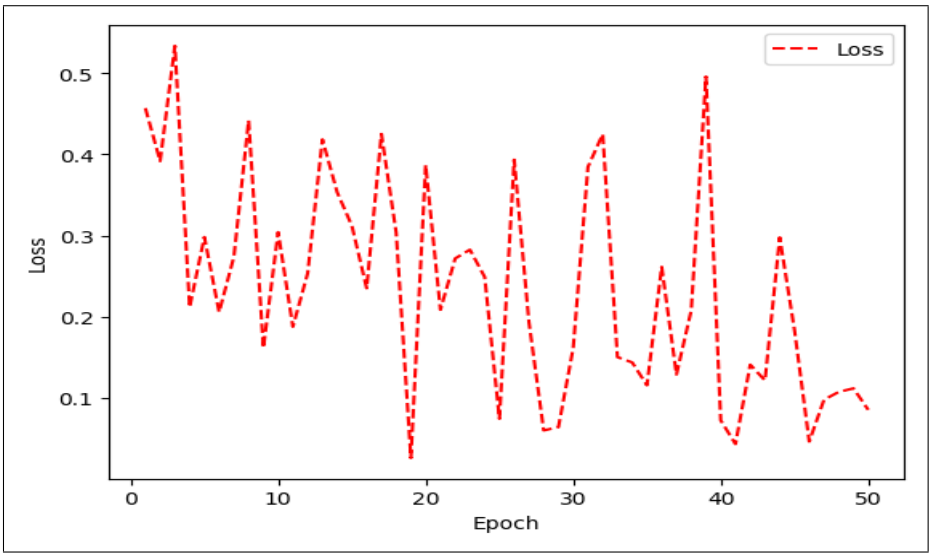

FIGURE 4.2: Negative Log Likelihood Loss for training data after 50 epochs upon retraining

After creating the separate kernels, we iteratively prune 75% of the neurons in each hidden layer, keeping the input and output layers untouched. To prune the neurons, we set all of their incoming and outcoming weights to 0. After pruning, we test our new subnetwork's accuracy. Additionally, we retrain this network to compensate for the lost accuracy. This is where our approach could edge out the other studies like (Frankle and Carbin, 2018), (Renda et al., 2020), (Gale et al., 2019), (He et al., 2018), and (de Jorge et al., 2020). Since we would drop entire neurons, the matrix used for calculations would be dense and hence more optimised. It is worth noting that Renda et al. reported a significantly higher cost for such retraining when not pruning entire neurons, so such a fine-tuning metric is not optimal in a sparse matrix.

For retraining, while (Frankle and Carbin, 2018) sets the weights to their original initialisation, we prefer to use the final weights from the initial training. This technique was shown to be effective by (Renda et al., 2020), and combined with our DPP approach, the pruning would be more efficient and accurate. We let the neural network retrain for the same number of epochs as our initial training, for 50 epochs. As we could see in the figure Fig 4.2, the loss function stabilises by the 50^{th} epoch.

Results

In our experimentation, we obtain three different accuracies. The first accuracy is reported by our fully connected original model. The next accuracy is reported by the pruned network, which has not yet been retrained. And the final accuracy is reported by the pruned and retrained neural network. We test our accuracy numbers for all three models on the 9984 test samples provided by the F-MNIST dataset. We shall look through each of these numbers.

5.1 Fully Connected Network

Upon testing, we obtained a net testing loss of 12.933 and correctly evaluated 8778 test samples. This brings our accuracy to:

$$8705/9984 = 0.8719$$

As described in Model Architecture and Dataset (2), we chose a batch size of 32. Figure 5.3 shows the batch negative log-likelihood loss function value vs. the batch number.

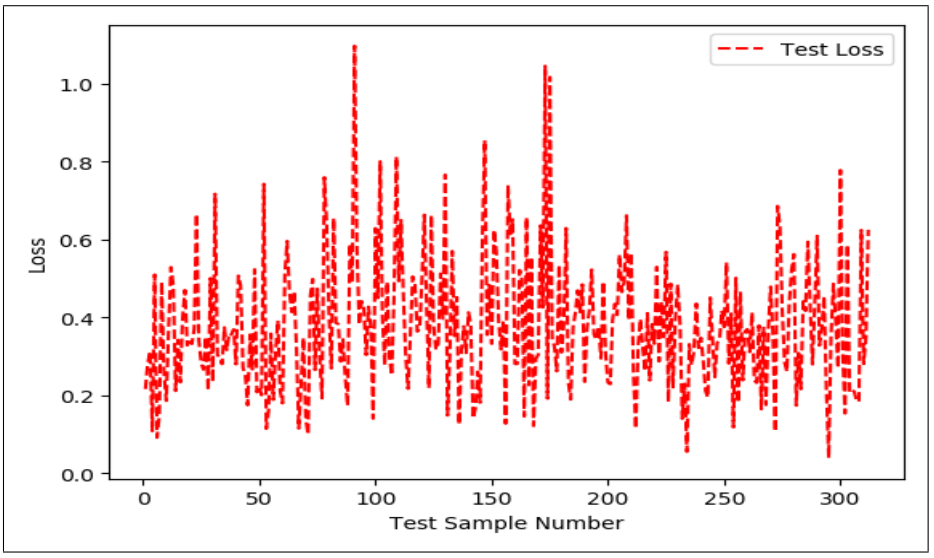

FIGURE 5.3: Batch loss vs. Batch Number

5.2 Pruned but Untrained Network

When we tested our pruned network, we observed a net testing loss of 31.258, and we could only correctly evaluate 6683 out of the 9984 samples. Hence our accuracy drops to:

$$6683/9984 = 0.6693$$

Figure 5.4 shows the batch negative log-likelihood loss function value vs. the batch number.

5.3 Pruned and Retrained Network

After pruning the network, we retrain our model and test its accuracy. Upon retesting, we observe a net testing loss of 12.180 and are correctly able to identify 8778 of the 9984, bringing our accuracy to:

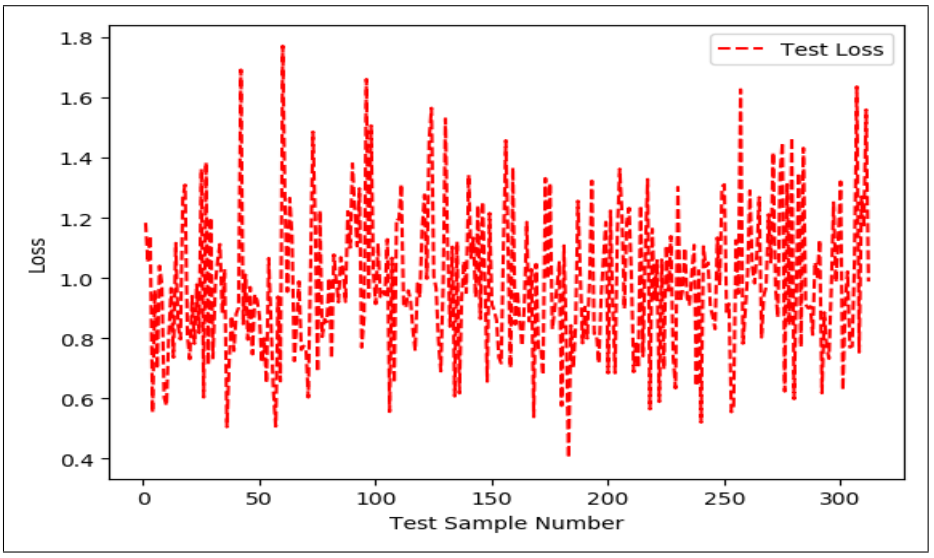

FIGURE 5.4: Batch loss vs. Batch Number

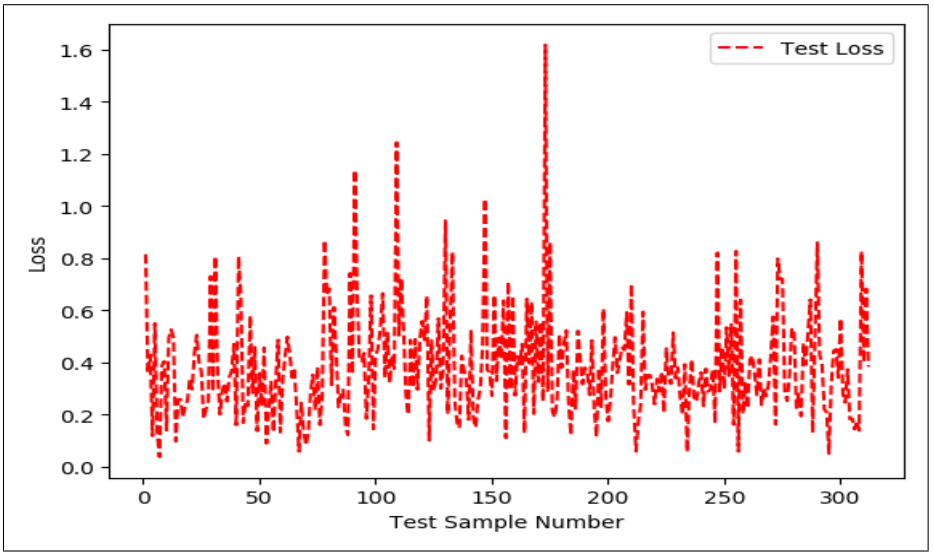

FIGURE 5.5: Batch loss vs. Batch Number

$$8878/9984 = 0.879$$

. Yet again, Figure 5.5 shows the batch negative log-likelihood loss function value vs. the batch number.

5.4 Key Takeaways

Upon our observations, we can clearly state that pruning the neural network does result in a drop in accuracy. While it does show promising results for a network with only 25% of its original strength, a 20% drop in accuracy could be dramatic, depending on the industry employing the model. Since our purpose is to develop an efficient pruning method to make complicated neural network models more accessible for all, we would like to reclaim this lost accuracy.

The work set up by DPP finally comes to fruition when we retrain our network. Upon retraining, we could see that the model now performs at par with the original model, if not better. We have been able to successfully prove that our method works for this particular type of architecture and that a *winning ticket* does, in fact, exist for a 25% sparse network.

Future Work

While we could only experiment with a single form of architecture this semester, we were able to prove that our methodology shows promise. Hence now that the pipeline is set, we will be conducting several experiments over the next few months.

First and foremost, we need to ensure that our training time has reduced. Ultimately, our pruning method aims to transform the complicated neural networks into a version that is much more accessible in terms of distribution and ability to run. An obvious way to confirm our success is to ensure that the training time for the model has been reduced from its original training time.

Secondly, since we were only able to implement our pruning technique on a single dataset of a single architecture (2), we would like to experiment with both of these parameters. We would like to take on several other datasets like MNIST, CIFAR-10, and CIFAR-100. We are also interested in extending this technique to several different neural network architectures like CNNs or RNNs.

Further, it would be intriguing to see if a similar number of epochs are required while retraining or if we could do away with fewer epochs. This could increase our efficiency. Also, we want to see the limit to which our pruned models give acceptable accuracies. Frankle and Carbin showed that even a 95% pruned network could match the accuracies of a fully connected model, and we aim to replicate the same. We could also play around with the kernels of DPP (2) and compare their results.

Bibliography

de Jorge, P., Sanyal, A., Behl, H. S., Torr, P. H. S., Rogez, G., and Dokania, P. K. (2020). Progressive skeletonization: Trimming more fat from a network at initialization.

Desai, S., Zhan, H., and Aly, A. (2019). Evaluating lottery tickets under distributional shifts. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 153–162, Hong Kong, China. Association for Computational Linguistics.

Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. (2019). Stabilizing the lottery ticket hypothesis.

Gale, T., Elsen, E., and Hooker, S. (2019). The state of sparsity in deep neural networks.

Han, S., Pool, J., Tran, J., and Dally, W. J. (2015). Learning both weights and connections for efficient neural networks.

Hayou, S., Ton, J.-F., Doucet, A., and Teh, Y. W. (2020). Robust pruning at initialization.

He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Kulesza, A. (2012). Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286.

LeCun, Y., Denker, J., and Solla, S. (1989). Optimal brain damage. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. (2016). Pruning filters for efficient convnets.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2755–2763.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. (2020). Proving the lottery ticket hypothesis: Pruning is all you need. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6682–6691. PMLR.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference.

Morcos, A. S., Yu, H., Paganini, M., and Tian, Y. (2019). One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers.

Mozer, M. C. and Smolensky, P. (1988). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress.

Renda, A., Frankle, J., and Carbin, M. (2020). Comparing rewinding and fine-tuning in neural network pruning.

Wang, C., Zhang, G., and Grosse, R. (2020). Picking winning tickets before training by preserving gradient flow.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. (2019). Deconstructing lottery tickets: Zeros, signs, and the supermask.