International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249 – 8958 (Online), Volume-9 Issue-4, April, 2020

Novel Pruning Techniques in Convolutional-Neural Networks

Rahul, Hritik Dahiya, Divyansh Singh, Ishan Chawla

Abstract: Deep Learning allows us to build powerful models to solve problems like image classification, time series prediction, natural language processing, etc. This is achieved at the cost of huge amounts of storage and processing requirements which are sometimes not possible in machines with limited resources. In this paper, we compare different methods which tackle this problem with network pruning. Selected few pruning methodologies from the deep learning literature were implemented to display their results. Modern neural architectures have a combination of different layers like convolutional layers, pooling layers, dense layers, etc. We compare pruning techniques for dense layers (such as unit/neuron pruning, and weight Pruning), and convolutional layers as well (using L1 norm, taylor expansion of loss to determine importance of convolutional filters, and Variable Importance in Projection using Partial Least Squares) for the image classification task. This study aims to ease the overhead in terms of optimization of the model for academic, as well as commercial, use of deep neural networks.

Keywords*: *Deep learning, Neural networks, Pruning deep networks, convolutional neural networks.

I. INTRODUCTION

Deep learning has been on rise for quite a while and has proved itself to be state-of-the-art for many real-life problems related to texts, images, audio, video etc. Researchers have achieved unexpectedly better results in various tasks such as image classification, image segmentation, object classification and detection, text mining, recommendation engines, speech recognition, prediction modelling and many others by employing deep learning with their approaches. The performance improves as we train deeper and deeper networks. For Example, on the Image classification task bigger and deeper architectures like InceptionResNetV2[23] (164 layers deep), Xception[1] (71 layers deep), ResNet50[8] (50 layers deep), InceptionV3[24] (48 layers deep), VGG16 and VGG-19[22] (16 and 19 layers respectively) etc. reduced the top-1 and top-5 error by a significant margin than the previous state-of-the-art architectures such as AlexNet[14] (8 layers) and LeNet[15] (6 layers deep). Since every moon has a dark side, this performance also comes at the price of computation as well as storage cost.

Even with the considerable speedup advantage of GPUs, it takes a significant amount of training time for deep networks. Big storage and computational requirements of these networks limit their usage to resource intensive machines, and makes it difficult for them to be deployed on mobile devices. As the network grows deeper, the number of parameters grows large as well. They tend to explode to the extent of redundancy, leading to overfitting and ultimately poor generalization capacity. Most intuitive answer to this issue is the removal of parameters which are redundant. This is what Pruning does and which leads to network compression and acceleration without hurting the performance considerably. It provides the solution to the computation issue by reducing the amount of expensive operations (FLoating point OPerations, or FLOPs[19]) as well as the storage issue by removing the redundancy. In this paper we discuss various pruning methods which are useful in reducing memory requirements and computational requirements of different models.

Section II of this study jots down some previous research related to our topic. Section III describes the pruning techniques, the implementation details are summarized in Section IV, and their results are noted and compared in Section V. At last, the research is concluded in Section VI of this study.

II. RELATED WORK

Accelerating deep networks has been a continuously researched topic for quite some time. Many researchers have been studying different kinds of pruning and tried to classify them into high level categories, like [21] categorised pruning algorithms as one of the two broad classes. One of them estimated the effect of removal of a weight, filter, channel or layer, depending on the granularity of pruning (sensitivity of the element) only to filter them out. The other group, on the other hand, modifies the objective function to suppress the unnecessary weights while training. Reference [20] took a step further and categorised the neural network compression techniques into one of three high level groups in their survey - Precision Reduction i.e. change the representation of weights(e.g. Quantization[2],[6], encoding etc.), Compact Network Design (e.g. SqueezeNet[11], MobileNet[10], [11] etc.), and Network Pruning.

In recent years, [9] proposed a two-step iterative approach for pruning a single layer, which is extended to multi-layered architecture as well. In the first step, the most representative channels are selected using LASSO (Least Absolute Shrinkage and Selection

Revised Manuscript Received on April 25, 2020.
* Correspondence Author

Rahul, Hritik Dahiya*, Assistant Professor in the Department of Computer Science and Engineering in Northern India Engineering College affiliated to GGSIPU, Delhi

Divyansh Singh, Pursuing B.Tech, Department of Computer Engineering at Delhi Technological University, India.

Ishan Chawla, Pursuing B.Tech, Department of Computer Engineering at Delhi Technological University, India.

Operator) regression and the redundant ones are removed.

The output of the unpruned network is reconstructed as closely as possible using the least-squares error in the second step, and the steps are iterated until the required sparsity level is achieved. Reference [17] introduced a scaling factor (γ) for the output of every convolutional channel, trained these factors alongside the network weights but with an additional sparsity loss imposed on the former (i.e. the loss function now includes an additional term which is a function of γ but independent of the network weights). γ is used as the criteria to identify which channel contributes more to the output and thus should be kept or pruned. The variance/scale factor of Batch Normalization Layer[12] (which follows the convolution layer) was considered to be an appropriate choice for γ. The authors from Nvidia provided a brute force approach to the pruning problem known as Oracle Pruning in [19]. They formulated pruning as an optimization problem of choosing a subset of parameters (here filters, not individual weights of every filter) which will lead to the minimum change in the objective function i.e. measure the effect of removing each non-zero parameter one by one, and then prune the ones with minimum effect on the objective function, for as many iterations needed to achieve the desired sparsity. It is very expensive computationally and requires lots of GPUs, hence it was treated as the baseline to check the correctness of filter ranking criterias.

Reference [18] introduced ThinNet, a framework for layer by layer filter pruning, which uses statistical information from the next layer to select the filters to be removed in the current layer. For every layer, a greedy algorithm is used for channel selection. Channels of the filters and feature maps of the previous layer which correspond to the filters to be pruned are also removed. After channel selection for a layer, the channels are weighed by the weights of a least square regression, which turns out to be a great initialization for the next step i.e. fine-tuning. These steps are iterated for the next layer. Reference [3] introduced GSM SGD (Global Sparse Momentum in stochastic gradient descent) method to prune deep networks. This method introduced a pruning technique which pruned the model during training, so repeated fine-tuning of the model post pruning is not necessary. The importance values for each layer is calculated using the taylor expansion of the loss function and then at each gradient update step, the least important weights are updated using only the regularization parameter and not the gradient update. This modified version of momentum SGD (Stochastic Gradient Descent) was termed as GSM SGD. This ensured pruning of filters or weights which did not gain importance during the training phase. This method is beneficial in many scenarios as it can work on small architectures as well. It is a model independent method which can be used for both filter and weight pruning.

Reference [4] proposed a hypothesis which states that a large network contains within itself a sub-network (W_0) which, when trained in isolation, can achieve the same (or even exceed) test accuracy of the unpruned network (W) in at-most the same number of training iterations as the original model. This sub-network is termed as the “Winning Ticket”. This winning ticket is sensitive to the learning rate. The authors improved upon their original theory in [5] so that it could be scaled to larger networks and bigger datasets as well. They eliminated the need for warm-up and introduced Late-Resetting i.e. instead of reinitializing the weights of winning ticket to that of iteration 0 (W_0), the network is reinitialized to the weights after a few iterations of pruning i.e. iteration i (W_i , i is a very small percentage of the total number of iterations, around 1%. Authors of the paper used i=1000 i.e. weights of 1000th iteration from the total of 112,000 iterations, for a VGG-19 network on CIFAR-10[13]). The winning ticket weights by late-resetting were found to be closer to those of the final pruned network (W_F) than the weights of winning ticket without late-resetting (closeness metric used was the cosine of the angles between W_F - W_0 and W_i -W_0).

III. PRUNING TECHNIQUES USED IN THIS STUDY

A. Pruning of dense neural networks

Dense Neural Networks are Fully Connected (FC) networks where each neuron of a layer is connected to every other neuron of the following layer. For the neural network to make a classification we only require a few selected connections. Hence, it is concluded that most of the connections in a dense network are redundant. Redundant connections can be pruned using unstructured pruning algorithms which do not depend on a particular layer or channel.

1) Weight pruning

The connections between the neurons can be ranked on the basis of their weight connections[7]. The weights with significantly low value do not affect the output of the neural net. Connections can be deleted where the weights are set to zero in the weight matrix. To achieve sparsity of s% in a network, we rank the individual weights in the weight matrix, W, according to their magnitude, and subsequently set the smallest s% weights to zero.

2) Unit/Neuron pruning

Instead of looking at an individual weight in the weight matrix, [7] removed the complete neuron along with all of its incoming and outgoing weights. Least important neurons are deleted. The entire column is set to zero in order to delete a whole neuron from the network. To achieve s% sparsity, columns of the weight matrix are ranked according to their L2 Norm and the smallest s% columns are removed.

B. Pruning of convolutional neural networks

Although a dense layer is heavily parameterized than a convolutional layer, because of parameter sharing, a major percentage of the total FLOPs occur in the convolutional part of the network. Thus, instead of pruning the dense layer, convolutional pruning provides a greater acceleration benefit. In the following subsections, convolutional pruning techniques compared in this study are introduced.

1) Smallest L1-norm pruning

Reference [16] discussed structured filter pruning of ConvNets using the L1 norm of the filters as the selection criteria. The gist of the pruning algorithm for a single convolutional layer is as follows.
The filters are sorted according to their L1 norm, after which s% of

the smallest valued filters are removed. The corresponding feature maps, as well as the corresponding filters of the following layer, are also removed. The ideology behind this criteria seems to be that smaller the filter values, smaller will be its contribution to the output. This technique can be implemented in either a local or a global fashion. When implemented, global pruning (i.e. when all the filters of the network are ranked together) broke down early in the pruning process (around 30% sparsity) for the smallest L1-norm filter selection criteria. At the break-down point, all of the weights of a layer are pruned and the network loses its learning capability. It performed no better than a simple probability calculator. Local pruning was implemented with a little variations i.e. Layer by Layer One-shot pruning and Layer by Layer Iterative fashion.

a) Layer by Layer One shot Pruning

For any sparsity level, say s, s% filters were removed from each convolutional layer in one shot. The network was fine-tuned to achieve the accuracy used for comparison in the result section.

b) Layer by Layer Iterative Pruning

Unlike one shot pruning, s% filters are not pruned all at once, but in baby steps. If the network is already t% pruned (t<s) and well adjusted to this level of pruning (i.e. fine-tuned), it can recover much faster than the one which is pruned in one shot for the same level of sparsity. It also tends to be more robust.

2) Taylor expansion of loss function

In this method, [19] took into consideration the value of loss/objective function, if any given filter to zeroed i.e. pruned. Let $L(D|\theta)$ be the loss function, where θ is the collection of all network parameters i.e. filters of every layer of the network for the input image set, *D*. If any filter *f* is zeroed out, then the change in loss function is given by (1).

$$\partial L = |L(D|\theta) - L(D|\theta, f = 0)| \quad (1)$$

Using taylor expansion and ignoring higher order terms, *dL* is estimated to (2).

$$\partial L = \left| f \times \frac{\partial L}{\partial f} \right| \quad (2)$$

For any filter, its importance value is calculated by taking the L1 norm of the filter across all channels i.e.

$$I(f) = \frac{\sum_{H}\sum_{W}\sum_{C} dL}{H \times W \times C} \quad (3)$$

where H, W, and C are the height, width and number of channels of the current filter respectively. The filters are pruned based on their importance values, layer by layer or globally. The amount of pruning or sparsity is a user-specified parameter. After this ranking of filters, following approaches for pruning the low ranking filters can be employed.

a) Layer by Layer Pruning

Here s% of the total number of filters are pruned for every convolutional layer, as if they were isolated from one another, according to their rank calculated using equation 3.

b) Global Pruning

Here s% of the total filters are pruned from the pool of filters, irrespective of the layer in which they reside.

c) Global Iterative Pruning

Here s% of the total filters are pruned on the basis of their global ranking, but instead of pruning the network in one go, the network is pruned in small steps where *k*% filters are removed at each step followed by fine-tuning. This process is repeated for *s/k* total steps.

3) Pruning filters using Partial Least Squares

Reference [26] discussed pruning of deep convolutional networks using Partial Least Squares. This algorithm estimates the importance of a filter on the basis of its relationship with the class label, and projects the same in a low dimensional space. It is an iterative algorithm where s% of the least important filters are discarded after each iteration. There are 4 phases of this algorithm:

a) Filter Representation

The filters of the convnet are represented in terms of feature vectors. This is realized using a simple pooling operation over the filters. Pooling operation results in either a single feature (when global pooling operation is used) or a set of features (when max-pooling 2 × 2 is used). The filters are represented in a high-dimensional space of dimensionality d.

b) Feature Projection

The high dimensional feature space is projected onto a low dimensional space of dimensionality c (where c << d) using the Partial Least Squares method. This latent space is represented by the feature vectors with maximum covariance with the class label.

c) Filter Importance

The filters are scored using Variable Importance in Projection (VIP) technique[25].

d) Prune and Fine-tuning

The set of scores for filters, {f1, f2, ..., fj}, is generated and s% (pruning ratio) of the filters are discarded according to their scores.

IV. IMPLEMENTATION DETAILS

A. Details for Dense networks

MNIST dataset was chosen for classification of 10 object classes with 60,000 images as training data and 10,000 images for testing. Each data sample is of 28 by 28 pixel grayscale images.

A simple feed-forward neural network with 4 hidden layers having 1000, 1000, 500 and 250 neurons in the corresponding layers is employed. Each hidden layer is followed by the ReLu activation function. Finally the output layer has the softmax function which classifies the output from hidden layers into 10 classes. The model is not fine-tuned post pruning.

B. Details for Convolutional networks

CIFAR-10[13] dataset was chosen for classification of 10 object classes with 50,000 images as training data and

10,000 test images. Architecture of the network used for the purpose of this study is VGG-16[22] model which was pre-trained on CIFAR-10. Majority of the techniques used in this paper were implemented on VGG-16. However, due to the complexity of the PLS algorithm and unavailability of computation resources, PLS was implemented on a smaller convolutional network having two Conv2D Layers with ReLu and a MaxPooling Layer, followed by another set of two Conv2D Layers with ReLu and a MaxPooling Layer. The activations are flattened. A Dropout Layer, and two FC layers with ReLu activation follow. Finally the output layer has a Softmax function which classifies the input into 10 classes. Model is fine-tuned for 10 epochs after each iteration of pruning to measure the pruning performance on the test set.

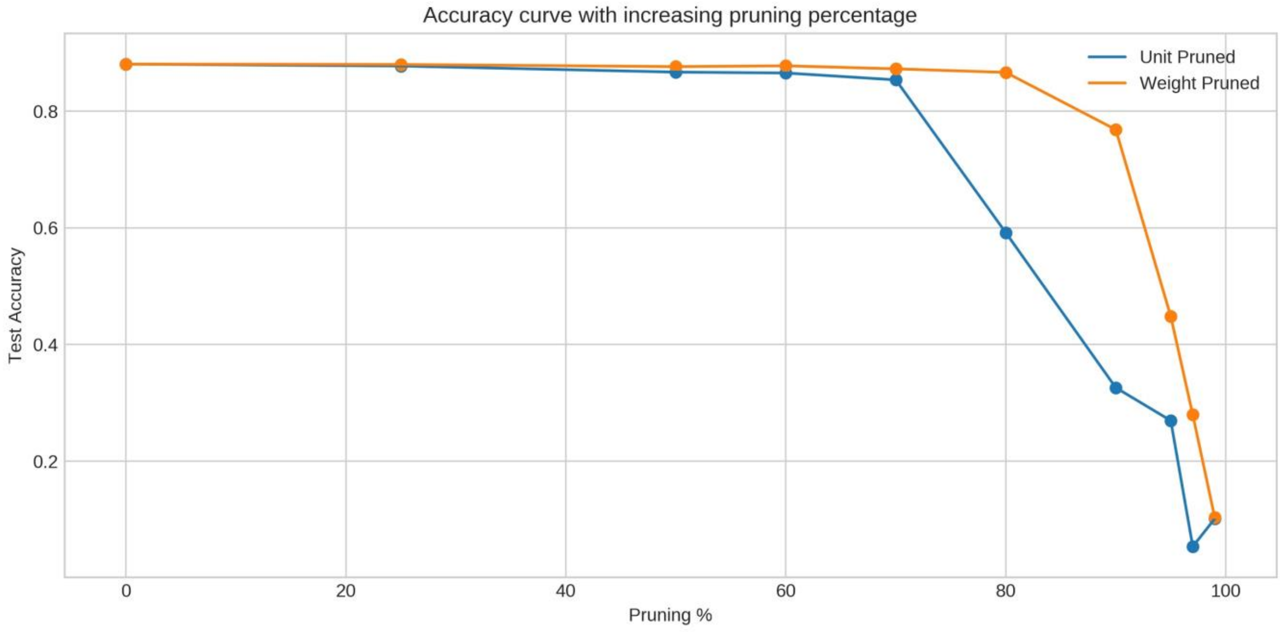

Fig. 1 Accuracy of unit and weight Pruned model with increasing sparsity

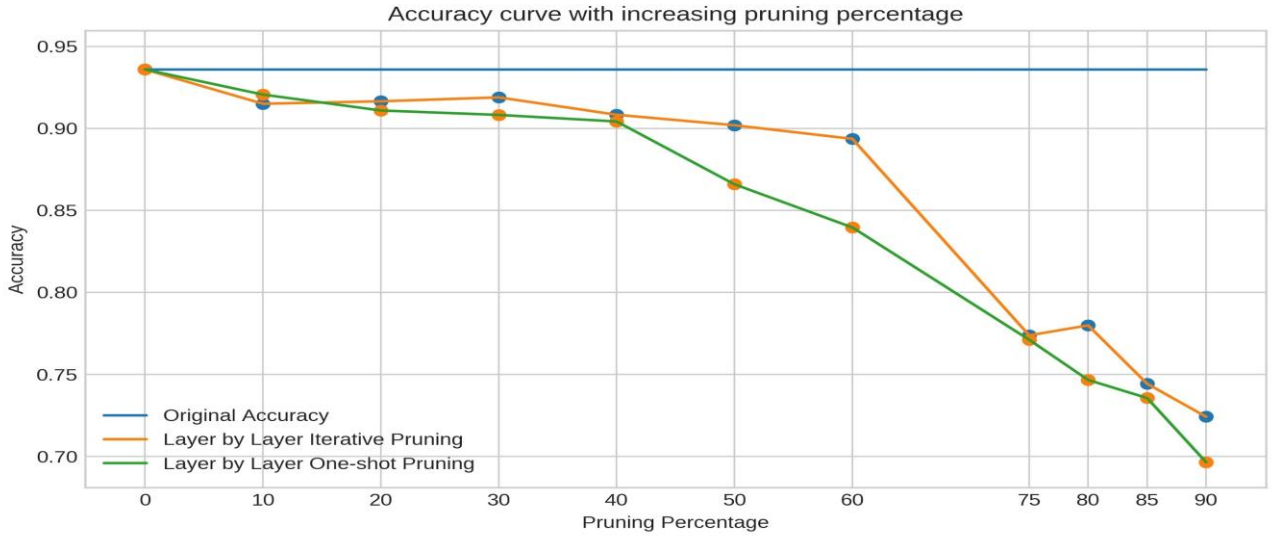

Fig. 2 Accuracy of local One-shot and local Iterative Smallest L1 norm pruned model with change in pruning percentage.

V. RESULT

A. Dense networks

Fig. 1 compares the performance of unit pruning and weight pruning of a feedforward dense network. It shows the change in accuracy when redundant neurons and redundant weights are removed with increasing sparsity in the network. In weight pruning, the test accuracy remains the same till 90 % sparsity, however it drops tremendously after that. Thus, it can be concluded that 90% of the weights are redundant in our model and it requires only 10%-15% of its initial weights to perform its task. Model performs faster and still achieves good accuracy when it is compressed. It is represented using less space, and the time complexity of the model is reduced by a significant extent. Given that the model is not fine-tuned after pruning, the model can perform even better if it is re-trained with the pruned weights.

The breakdown point for Unit Pruning is at 70% sparsity. After which the model starts to disintegrate as more columns are removed from the weight matrix. It is concluded that 70% neurons can be pruned from this network without significant loss of accuracy. Therefore, weight pruning holds advantage over unit pruning in terms of the compression because it remains unaltered till 90% sparsity. Whereas, unit pruning is much faster than weight pruning due to its structured methodology. A combination of both techniques can provide good compression and acceleration benefits.

B. Convolutional Networks

Fig. 2 shows how the selected network reacts to pruning according to the smallest L1-norm criteria. The network after iterative approach is close to 90% accuracy at 60% sparsity, unlike one-shot fashion which is slightly less than 85% accurate Following which the accuracy drop becomes significant. Iterative approach clearly outperforms one-shot pruning at significant levels of sparsity.

Reaction of network to pruning by taylor expansion of loss is shown in Fig. 3. Result of global pruning are much better than that of layer-by-layer pruning in Taylor loss criteria and the accuracy holds itself to a reasonable level upto 40% sparsity i.e. after 1690 filters of the original 4224 filters are removed, the accuracy dropped from 93.59% to 91.91% (a drop of 1.68% on the test data).

This result looks good till the sudden drop in accuracy is observed at 50% sparsity. It is due to the fact that some layers get pruned in their entirety and the network is not able to recover from the damage. After 60% pruning, the model gets completely damaged.

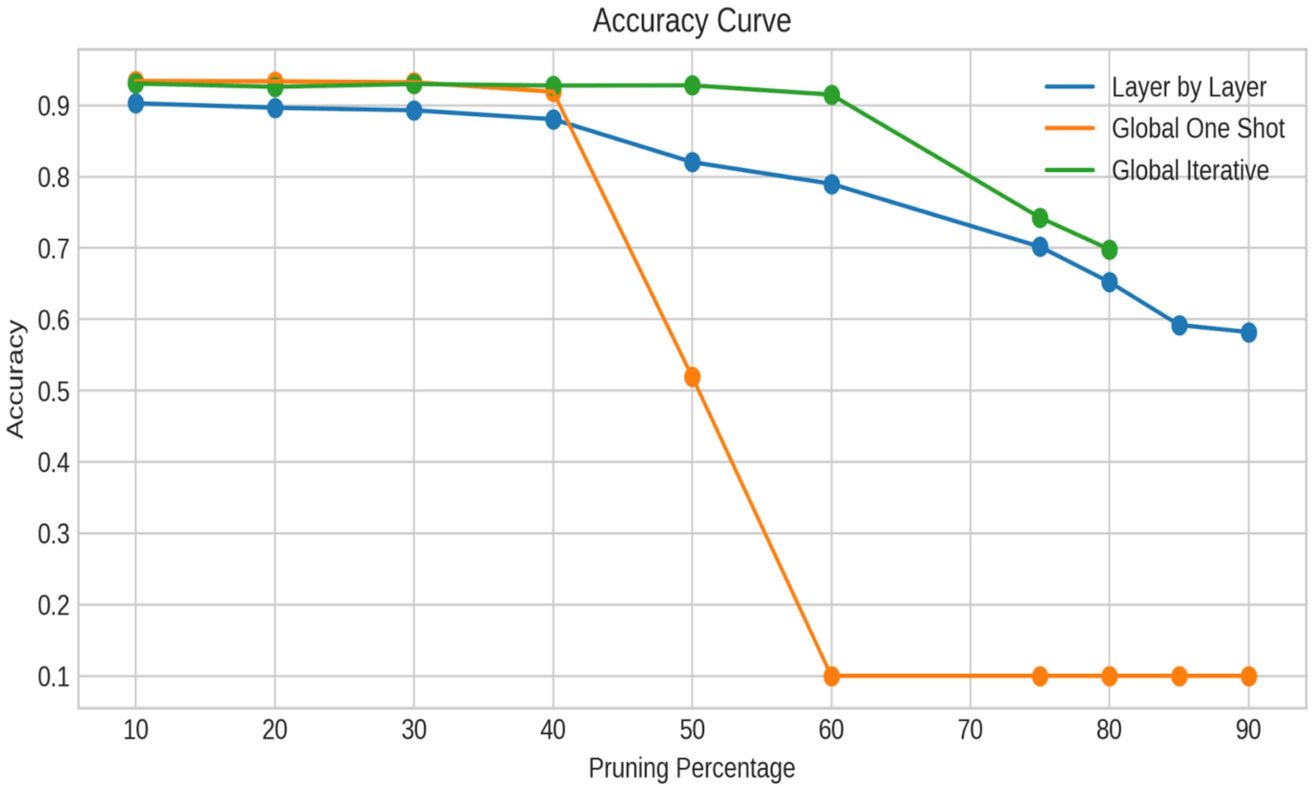

Fig. 3 Accuracy of local One-shot, global, and global Iterative taylor loss pruned model with change in pruning percentage.

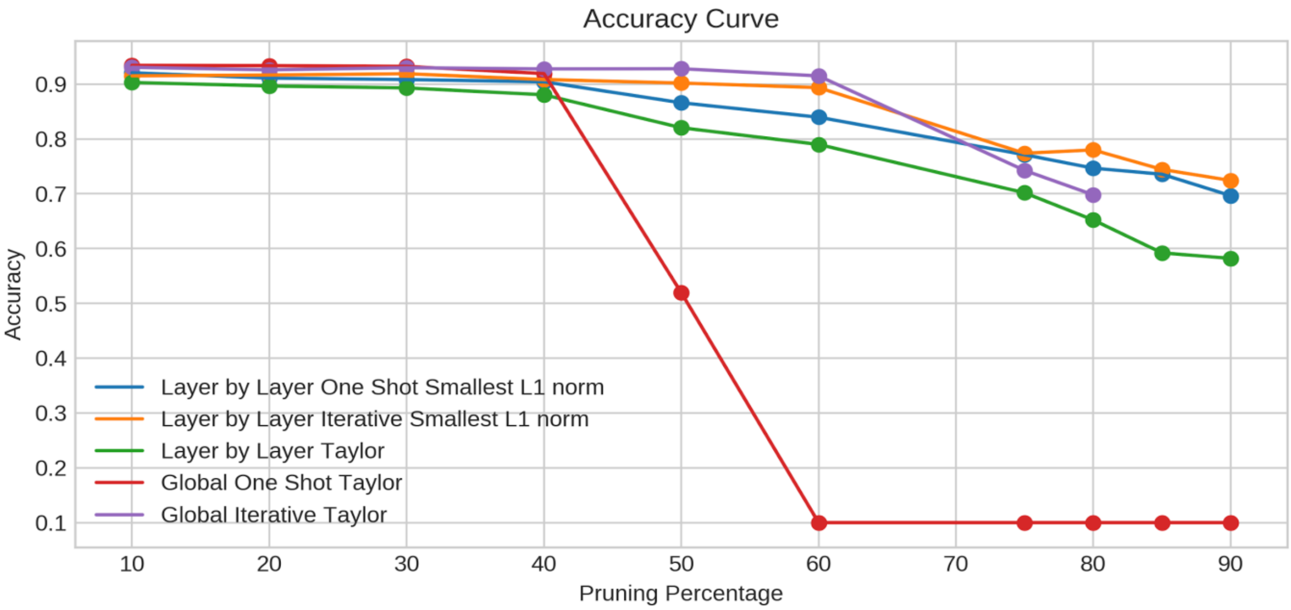

Fig. 4 Comparison of Accuracy of convolutional pruning techniques with change in pruning percentage.

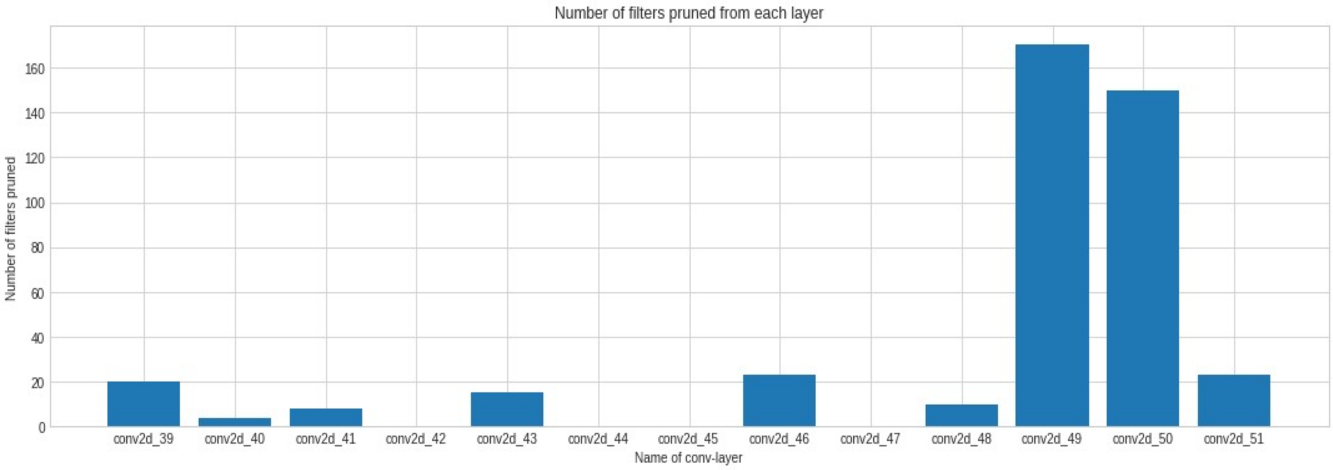

Fig. 5 Distribution of filters pruned in global iterative pruning using taylor expansion of loss.

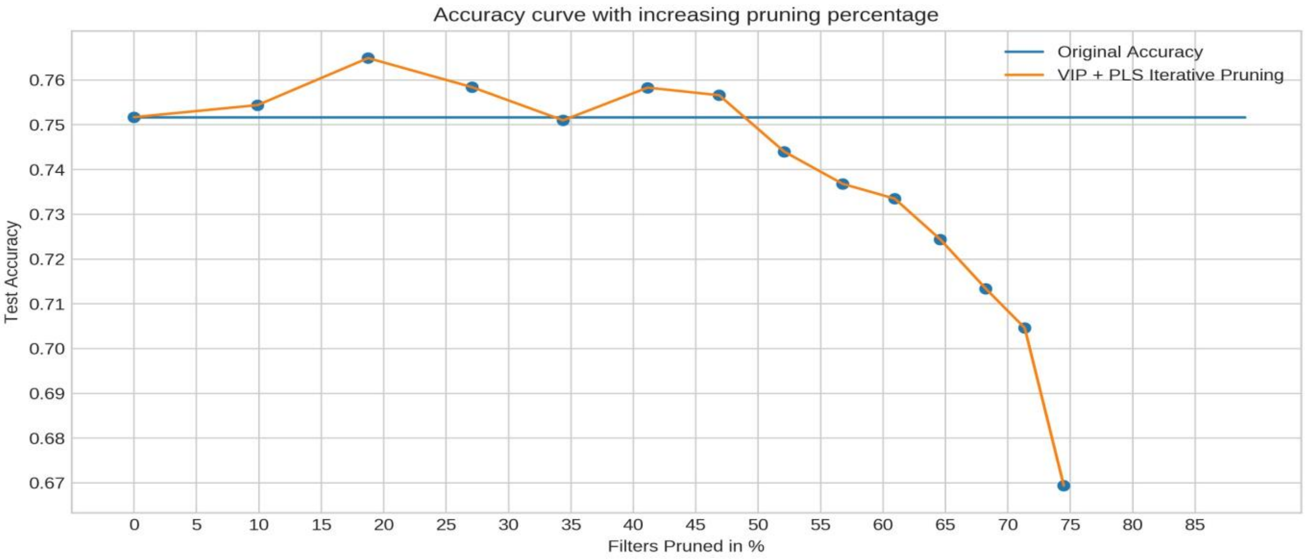

Fig. 6 Accuracy of the VIP PLS pruned model with percentage of filters pruned

The downside of the global iterative method is the amount of time needed to train, when fine-tuning was included in the process. In an ideal case with unlimited computational resources, the network can be fine-tuned after removal of a single filter and till the required sparsity is achieved, but this is not practical. Hence, a trade-off between the size of the step and the final accuracy achieved is necessary. In this study, a step size of 10% pruning was taken i.e. at each step around 423 filters are removed before fine-tuning.

Pruning techniques applied to the VGG model are compiled and compared in Fig. 4 according to accuracy drop. At the breaking point, clearly iterative approaches take the crown in terms of accuracy retention.

FLOPs reduction was also used as a metric beside accuracy in order to evaluate the performance of pruning techniques. Calculation of number of FLOPs in any convolutional layers was provided in [19] using the following formula:

$$FLOPs\_conv = 2 \times H \times W \times ((C_{in} \times K^2) + 1) \times C_{out} \quad (4)$$

where H, W, C_{in}, K and C_{out} are height, width of the input feature map, input channels ,kernel size and output channels respectively. FLOP reduction was used to evaluate the performance advantages of the above selected iterative algorithms, namely Layer by Layer Iterative Smallest L1 Norm pruning and Global Iterative Taylor Loss criteria, which are described in Table I. The values in the table correspond to the points where the accuracy drops sharply, which appears to be at 60% sparsity in Fig. 4.

Table I: Comparison of Filter selection criterias at 60% sparsity

	Performance of techniques		
Pruning Method	FLOPs Reduced	Filters Pruned	Accuracy Drop
Smallest L1 Norm	102,075,728 (34.42 %)	2531	4.23%
Taylor Expansion of Loss	133,813,416 (45.12 %)	2512	2.1 %

Fig. 5, which shows the distribution of unimportant filters when pruned globally according to taylor loss criteria, and Fig. 4 together provide an important observation. Fig. 5 concludes that the majority of redundant weights lie in the later or deeper layers of the network. Earlier weights allow the network to capture high level details (e.g. shapes, lines, textures etc.) for the classification task at hand, and thus add more value to the decision making process. Hence, when the accuracy takes a sharper dip at 60% sparsity, Taylor criteria holds the highest accuracy among others (refer Fig. 4).

Results of the networks after VIP PLS pruning can be seen in Fig. 6. There is not much drop in accuracy when the pruning percentage is increased; accuracy remains consistent until 70% of the filters are removed. The steep fall in accuracy at 70% pruning is the breaking point of the model. Accuracy even increases a little compared to the initial accuracy until the 50% pruning mark.

Let's see the importance of fine-tuning. Fine-tuning after pruning retrains and tunes the model keeping in context of the lost connections, filters or neurons depending upon the pruning technique and the ranking algorithm. Retraining after pruning is a must as it enables the model to classify better after some classifying power is lost due to pruning. It is visible how the accuracy curves of the techniques differ. PLS maintains higher accuracy than the original till 50% pruning, whereas the other methods never gain a higher accuracy than the one they started with. Although it doesn't make sense to compare techniques implemented on different architectures, the upward trajectory of the PLS curve is enough to determine its superiority.

VI. CONCLUSION

In this study, the results observed by implementing various ranking algorithms in different ways to prune a deep network were showcased, along with their effects on the Floating Point Operations (FLOPs) without hurting the performance appreciably. The decision making process of selecting the desired approach is thus reduced to the trade-off between performance and the amount of pruning.

For Dense networks, Unit pruning provides simplicity and ease implementation over weight pruning, whereas the latter out-performs the former.

For Convolution Networks of low to medium complexity, VIP PLS technique outperforms the other techniques. But in a highly complex network the aforementioned technique lags behind. Moreover VIP PLS is extremely computationally expensive compared to other techniques. In general, Taylor Expansion of Losses criteria is much more suitable for complex networks and outperforms L1 Norm pruning in ranking the performance of filters.

Keeping the choice of pruning technique aside, different approaches to their implementation affect their performance i.e. either in a global or local fashion, and either in one-shot or a number of iterations. Global approach is free from the assumption that every layer has an equal number of redundant filters, unlike the local approach. But the global fashion falls victim to the breakage of the network when the pruning algorithm declares a large part of the layer redundant. Similarly, when the pruning technique is implemented using a single shot approach, it falls short in front of an iterative approach; since small steps to achieve a pruning percentage allow the network to recover from the removal of parameters in a step by step fashion.

Refining neural nets can also be beneficial for transfer learning since a bigger model pre-trained on a different but similar, bigger dataset can be used for a smaller problem and the redundant weights can be pruned to return a smaller and compact model with comparable accuracy to the original bigger model. Pruning of Deep Neural Networks, in future, can open the world of deep learning to small, mobile devices and reduce its dependency on devices with high-end computational and memory resources. Since, the crux of a large number of pruning methods is its ranking algorithm, development of new and better ranking methods which can make identification of redundant weights much more accurate is a path worth exploring in future research.

ACKNOWLEDGMENT

We are grateful to our mentor, the Department of CSE, Delhi Technological University for providing us with facilities and a platform to complete this study.

REFERENCES

1. F. Chollet, Xception: Deep Learning with Depth Wise Separable Convolutions, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017).
2. M. Courbariaux, Y. Bengio, J.-P. David, BinaryConnect: Training Deep Neural Networks with binary weights during propagations, arXiv [cs.LG]. (2015).
3. X. Ding, G. Ding, X. Zhou, Y. Guo, J. Han, J. Liu, Global Sparse Momentum SGD for Pruning

Very Deep Neural Networks, arXiv [cs.LG]. (2019).

4. J. Frankle, M. Carbine, The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, arXiv [cs.LG]. (2018).
5. J. Frankle, G.K. Dziugaite, D.M. Roy, M. Carbin, The Lottery Ticket Hypothesis at scale, arXiv [cs.LG]. (2019).
6. P. Gysel, M. Motamedi, S. Ghiasi, Hardware-oriented Approximation of Convolutional Neural Networks, arXiv [cs.CV]. (2016).
7. S. Han, J. Pool, J. Tran, W.J. Dally, Learning both Weights and Connections for Efficient Neural Networks, arXiv [cs.NE]. (2015).
8. K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016).
9. Y. He, X. Zhang, J. Sun, Channel Pruning for Accelerating Very Deep Neural Networks, Github, n.d.
10. A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv [cs.CV]. (2017).
11. F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE, (n.d.).
12. S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv [cs.LG]. (2015).
13. A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, (n.d.).
14. A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, (n.d.).
15. Y. LeCun, L. Bottou, B. Yoshua, P. Haffner, Gradient-Based Learning Applied to Document Recognition, (n.d.).
16. H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning Filters for Efficient ConvNets, arXiv [cs.CV]. (2016).
17. Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning Efficient Convolutional Networks through Network Slimming, 2017 IEEE International Conference on Computer Vision (ICCV). (2017).
18. J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, W. Lin, ThiNet: Pruning CNN Filters for a Thinner Net, IEEE Trans. Pattern Anal. Mach. Intell. 41 (2019) 2525–2538.
19. P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning Convolutional Neural Networks for Resource Efficient Inference, arXiv [cs.LG]. (2016).
20. R. Pilipović, P. Bulić, V. Risojević, Compression of Convolutional Neural Networks - A short survey, (n.d.).
21. R. Reed, Pruning algorithms-a survey, IEEE Trans. Neural Netw. 4 (1993) 740–747.
22. K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv [cs.CV]. (2014).
23. C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, arXiv [cs.CV]. (2016).
24. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the Inception Architecture for Computer Vision, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016).
25. T. Mehmood, K. H. Liland, L. Snipen, and S. Sæbø, “A review of variable selection methods in Partial Least Squares Regression,” *Chemom. Intell. Lab. Syst.*, vol. 118, pp. 62–69, 2012.
26. A. Jordao, R. Kloss, F. Yamada, and W. R. Schwartz, “Pruning Deep Neural Networks using Partial Least Squares,” 2018.

AUTHORS PROFILE

Rahul has done B.Tech in CSE and M.Tech in Information Security and currently working as an Assistant Professor in the Department of Computer Science and Engineering in Delhi Technological University, Govt of NCT of Delhi. He has previously worked as an Assistant Professor in Department of Computer Science and Engineering in Sant Longowal Institute of Engineering and Technology, Under MHRD, Established by Govt of India and also worked as an Assistant Professor in the Department of Computer Science and Engineering in Northern India Engineering College affiliated to GGSIPU, Delhi. He has many research publications in reputed International Conferences and Journals and having more than 4.5 years of teaching experience. His research interests are machine learning, deep learning, recommender systems, natural language processing, sentiment analysis.

Hritik Dahiya is pursuing his B.Tech. in Computer Engineering from Delhi Technological University. He has a review paper, “A Review of Trends and Techniques in Recommender Systems” published in the IEEE digital library. His research interests include, and are not limited to, Deep Learning, Computer Vision, and Machine Learning.

Divyansh Singh is pursuing his B.Tech. in Computer Engineering from Delhi Technological University. He has a review paper, “A Review of Trends and Techniques in Recommender Systems” published in the IEEE digital library. His research interests include and are not limited to Deep Learning, Computer Vision and Machine Learning.

Ishan Chawla is currently pursuing his B.Tech in the Department of Computer Engineering at Delhi Technological University, India. He has previously worked in integrating Deep Learning with Finance in Stock Sentiment Analysis, Earning and Bankruptcy Prediction. His research interests are in Deep Learning, Computer Vision, Machine Learning and Natural Language Processing.