Comparative Analysis of Channel Pruning of VGG-16 using L1/L2 Regularization

Abhishek Kumar
Information Technology
National Institute of Technology, Karnataka
Surathkal, Mangalore, Karnataka, India
ak.202it001@nitk.edu.in

Nitin Sharma
Information Technology
National Institute of Technology, Karnataka
Surathkal, Mangalore, Karnataka, India
nitinsharma.202it017@nitk.edu.in

Pragnesh Thaker
Information Technology
National Institute of Technology, Karnataka
Surathkal, Mangalore, Karnataka, India
pragnesh.187it001@nitk.edu.in

Biju R Mohan
Information Technology
National Institute of Technology, Karnataka
Surathkal, Mangalore, Karnataka, India
biju@nitk.edu.in

***Abstract*—Channel pruning is an extensively used approach that focuses on convolution layers of CNN based models for compressing a deep neural network. This approach focuses on selection of channels with the least importance, completely removing them from our model and then recompiling the model again for testing. In this paper, we are considering two channel selection methods i.e. lasso regression and ridge regression. The total focus of our work will be to reduce channels from the VGG-16 model using both these techniques. We have generalized the concept of single layer pruning and then extended it for multi layer as well as whole VGG-16 model pruning. Finally, we have done performance analysis of both ways of channel selection, keeping into consideration model structure and details before and after pruning. Upon observing results, we found that multi-layer pruning performs way better than the whole model pruning.**

***Index Terms*—VGG-16, Lasso Regression, Ridge Regression, Channel Pruning, Performance Evaluation etc.**

I. Introduction

CNN based deep learning models are been extensively used in image classification field. If we talk about today's time, we have very deep neural networks that are optimized for performance in terms of prediction accuracy. These deep neural networks are very huge and have parameters in the order of millions, making it problematic when exploring the scope of deployment of these models for real time use over IOT based devices, mobile applications, etc. as these huge models are computation intensive and requires a lot of resources but our mobile/iot based devices run on low power and have resource restrictions. So, the idea is to reduce the parameters of these deep neural networks keeping into consideration that there should be less reduction in accuracy and huge reduction in size and resource consumption.

There are various pruning methods already existing, with different use cases, for neural network compression and these methods are divided broadly into structured and unstructured methods. Some of these focus on reducing space required for saving model, while some focus on speed up while prediction. Structured simplification is a CNN based acceleration technique which mainly involves Tensor Factorization, Sparse connection and Channel pruning. Channel pruning for us is topic of focus here.

Channel pruning is a structural model compression technique used to reduce feature map depth. This method shrinks a network into a thinner one by reducing number of channels in each layer which leads to reduction in parameters as well as overall model size. To perform this we will focus on a two-step process. In first step we are going to focus on selection of channels to prune in such a way that accuracy gets least affected and in second step we reconstruct the output with remaining channels.

For selecting the channels we have considered two approaches i.e Lasso regression and ridge regression. After selection and elimination of these channels fine-tuning will be done. Finally we have done the performance analysis of the above two techniques to find which selection technique will lead to less reduction in accuracy and high reduction in size as well as parameters such that the model can accelerate.As evaluation metrics we are focusing on test accuracy, flops, size of model and parameter reduction.

II. Literature survey

Pruning can be widely categorized into two categories : Unstructured and Structured pruning. Unstructured Pruning(fig. 1) deals with removal of weight connections from a neural network by setting their values to 0. This introduces multiplication by 0 which can be ignore while testing. The model files thus created can be stored compressed on disk, taking up much less space. On the other hand, in Structured Pruning (fig. 2), we focus on removing group of parameters from model mostly in form of channels/filters. This results in the system becoming significantly faster as now lesser amount of FLOP (floating point operations) are to be preformed. We, in our work, are specifically focusing on Channel pruning and some relevant work done in that field are discussed below.

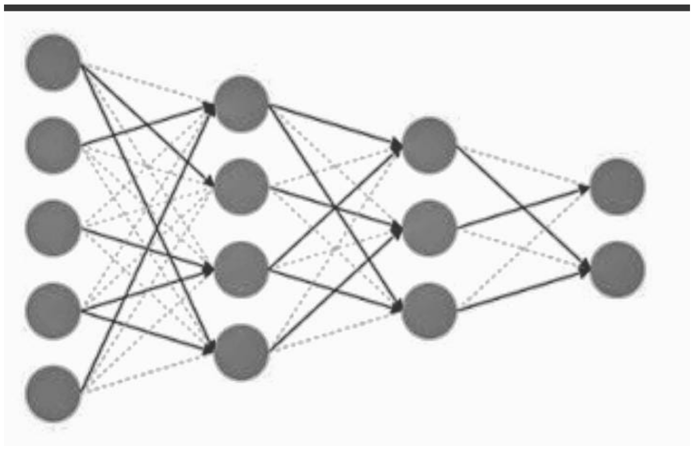


Fig. 1. Sample example for unstructured pruning

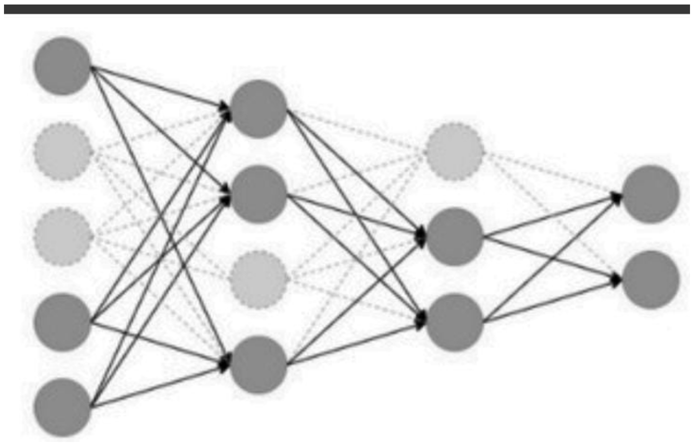


Fig. 2. Sample example for structured pruning

Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures[1] They introduced network compression where they iteratively optimized network by deleting neurons that are not important, Their approach removed low activation neurons which contributed very less to the final result without actually reducing accuracy by alot. They performed experiments on LeNet and VGG-16 achieving almost same accuracy with 2-3 times reduced parameters.

Play and prune: Adaptive filter pruning for deep model compression[2] They build a framework called Plan and Prune (PP) which uses an adaptive rate of pruning to prune and fine tune CNN based models while trying to preserve the models performance. Their model consisted of two steps where first step dealt with filter pruning and second one works towards maximizing accuracy while pruning. Their methods, instead of working on how much to prune, were focused on how much maximum error was allowed during pruning. Through this approach they were able to reduce parameters of VGG-16 model by 17.5 % and floating point operations by 6.4% with no relevant drop in accuracy.

Learning Efficient Convolutional Networks through Network Slimming [3] They worked on a slimming approach in which they took large deep learning models as input and impose sparsity induced regularization on the scaling factor in batch normalization layers resulting in identification of channels that do not contribute significantly to the output and then they perform pruning on it. They experimented with multiple datasets and achieved 20% reduction in model size and 5% reduction in computing operations for VGG-16.

Lossless CNN Channel Pruning via Decoupling Remembering and Forgetting[4] They proposed an approach in which they divided CNN into two parts : remember and forget. Remember part was used to maintain the model performance and forget part dealt with learning for efficiency. Their experiments were performed over Resnet-50 model and they were able to reduce number of floating point operations by 45% and with no accuracy drop. Their work was first to achieve lossless pruning with that high compression ratio.

III. Performance Metrics

This section gives insight on the methods for evaluating our pruning. Readings were taken before and after pruning of VGG-16 model for the following metrics :

1) **Test accuracy**: Test accuracy defines how accurately our model is predicting the result.
2) **Total Parameters**: Total parameters in our VGG-16 model.
3) **Model Size**: Space required in Megabytes to save our model.
4) **Flop (Floating point operations)**: Floating point operations in our model.

FLOP is closely related to time required during testing of a model as time is directly proportional to computations happening in a system. For calculating FLOP, Kerop library was used.

IV. Detailed Objective

Objectives of this work is to understand VGG-16 model structure and :

1) understand channel selection criteria using L1-Norm and perform multi layer pruning and whole model (Global Pruning) pruning on VGG-16 model.
2) understand channel selection criteria using L2-Norm and perform multi layer pruning and whole model pruning on VGG-16 model.
3) do comparative analysis between both channel selection techniques for multi and whole model pruning.

V. Approach

The main aim of the channel pruning is to reduce the width of the input to various layers so that the number of parameters and amount of computation could be reduced.

For instance, assume a conv layer where filter size=(f,f), input size from previous layer is (m x n x d) where m,n,d is length, height and width respectively, and k= number of filters. Now if we calculate the number of parameters i.e. f*f*d*k, number of computation f*f*d*(n-f+1)*(n-f+1)*k. So, we can see that both number of parameters as well as number of computation depends upon 'd' i.e. width. So by channel pruning we shall reduce the width so that the number of parameters and amount of computation both will also get reduced and since model size very much depends upon the number of trainable parameters, it will also get reduced. Moreover time required for classification of image is directly propotional to amount of computation happening during prediction.

Now the main question here is, how to decide which channel to be deleted. So for that we have to develop a channel selection mechanism which will be used to select the channels then deletion of those channels will be done. The two selection techniques are given as:

1) **Lasso Regression (L1-Norm)**: In simple words, it is the sum of all the coefficients of the channel.

$$L1 - Norm = \sum_{x \in channel'c'} x$$

2) **Ridge Regression (L2-Norm)**: In simple words, it is the sum of squares of all the coefficients of the channel.

$$L2 - Norm = \sum_{x \in channel'c'} x^2$$

In both lasso and ridge regression we have neglected biases, because we have to just check which channel is redundant and for that we do not need any extra dependency. After finding the most redundant channel we are going to delete them to reduce computation while testing. Once we are done with pruning, we need to reconstruct the output using the remaining channels and then perform prediction to test pruned model.

A. VGG-16 Model

This is a deep neural network which has been extensively used for image classification, object recognition, segmentation etc. tasks. It was first introduced in imagenet challenge 2014 where it was the winner. In this deep neural network model there are 13 conv layers and 3 fully connected layers present.

B. Dataset Insight

Our work is totally done over Image-net dataset which originally has 1000 classes in total and we created our VGG-16 model by using pre-trained weights for image-net dataset. For experimentation. we have used a subset of Imagenet dataset. We first load the pre-trained VGG-16 model which has been trained on image-net dataset then we re-compile the model and then test the accuracy of the model before pruning. After pruning the model we again recompile the pruned model and evaluate performance matrices again.

C. Multi-Layer Pruning

First we need to select the convolution layer at which we want to perform the compression and then decide how many channels need to be deleted at each convolution layer. Here, we are assuming that the rate of deletion 'r' will be similar throughout every convolution layer that we have selected. Now, after deciding the layers we will compute the L1-Norm or L2-Norm depending on which selection technique we are using. Then we will sort the scores/norms in ascending order and choose the first 'r' percent channels for that particular convolution layer. These selected channels are redundant channels because they are less representative and the rest of the channels have more importance in output than these. Then we delete these selected channels and retain the rest of them. Finally, do the fine-tuning i.e. reconstruct the output using left channels and repeat the same work across all the selected convolutional layers.

D. Whole model pruning

Here also we need to select the convolution layers at which we want to perform the compression then decide the rate of the pruning 'r' i.e. how many percentage of channels we want to delete across all the selected convolution layers. So for doing that we shall find the scores/norms(L1-Norm or L2-Norm) depending on the channel selection technique. We will do this step across all the layers rather than going layer by layer. Then sort these scores in ascending order and select the first 'r' percent channels from the sorted channels based on the scores. These first 'r' percent of channels will be the most redundant channels and rest are having more importance in finding output than these. So we delete the selected channels and finally we shall do the fine-tuning i.e. reconstruct the output using left channels.

Finally, we will do the performance analysis based upon certain performance evaluation parameters that we have discussed above. That means before applying any pruning techniques (multilayer pruning or whole model pruning) we have calculated all those parameters, then after doing pruning we have again calculated all those parameters and observed the effect of pruning. In this way we have analyzed at what value of rate we got the best performance as well as which channel selection technique is better between L1-regularization and L2-regularization. We have mentioned all these observations in the Observations and Conclusions section.

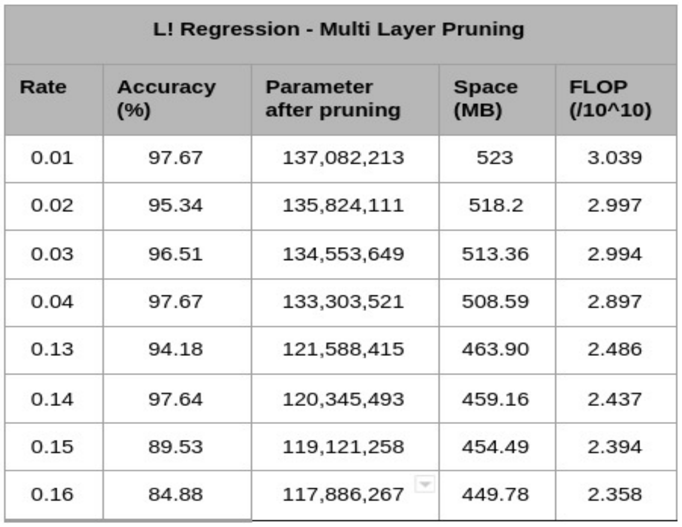
VI. Experiments and Results

We did our experiments on VGG-16 model using image-net dataset. VGG-16 is a 16 layer deep neural network with 13 convolution layers from where filters are to be pruned. It is widely used for image classification. We created a model with pre-trained weights for image-net and tested it before pruning over around 85 images belonging to different classes. Model before pruned had test accuracy of 98.8%, floating 138,357,544 total parameters, 30,390,582,674 floating point operations and required 528 MB of memory for storage. Pruning up to 0.05 loss i.e. 5% loss in accuracy has been considered.

Pruning of a single layer involves two steps. First step of selection of channel based on L1 or L2 regression while in second step we prune our model by removing those selected channels and then do prediction on test dataset. This concept in our work has been taken as base and extended to perform multi-layer and whole model pruning.

A. *Multi-Layer Pruning*

In this sub-section, we focused on how our models performed under multi-layer pruning. In case of multi-layer pruning, we are given a pruning rate ‘r’ which suggests that for each convolution layer, we have to prune r% of channels. These channels can be selected in one of two ways i.e. L1 and L2 based channel selection as discussed in above sections. Results for both of these methods are given in tables below in fig 3 and fig 4.

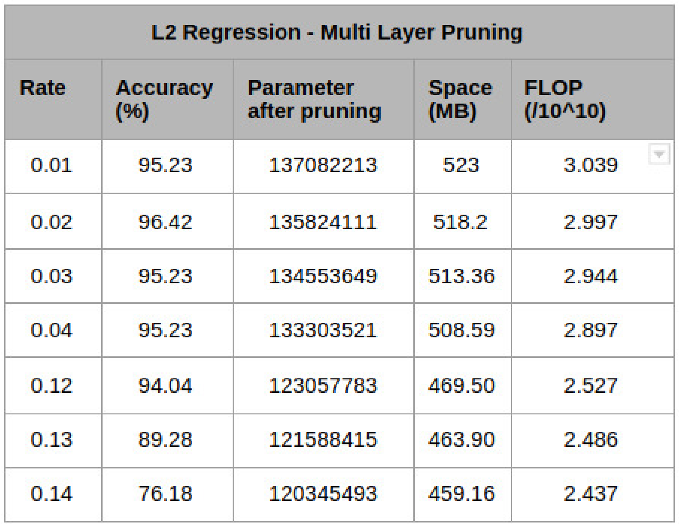


L! Regression - Multi Layer Pruning

Rate	Accuracy (%)	Parameter after pruning	Space (MB)	FLOP (/10^10)
0.01	97.67	137,082,213	523	3.039
0.02	95.34	135,824,111	518.2	2.997
0.03	96.51	134,553,649	513.36	2.994
0.04	97.67	133,303,521	508.59	2.897
0.13	94.18	121,588,415	463.90	2.486
0.14	97.64	120,345,493	459.16	2.437
0.15	89.53	119,121,258	454.49	2.394
0.16	84.88	117,886,267	449.78	2.358

Fig. 3. Readings of Multi-Layer pruning with L1-regularisczation channel selection strategy.

In case of pruning of channels selected by L1 regression, we were able to prune up to 14% pruning per layer without loosing more than 5% in accuracy. At this point, memory required to save the model reduced from 528 MB to 459 MB, floating point operation reduced by 22% and total parameters reduced by 14%. In case of L2 regression, we were able to prune up to 12% pruning rate per layer without loosing more than 5% in accuracy. At this point, memory to save model reduced to 469.5 MB, FLOP reduced by 19% and total parameters reduce by 12%. So it can be seen that L1 method outperformed L2 one.



L2 Regression - Multi Layer Pruning

Rate	Accuracy (%)	Parameter after pruning	Space (MB)	FLOP (/10^10)
0.01	95.23	137082213	523	3.039
0.02	96.42	135824111	518.2	2.997
0.03	95.23	134553649	513.36	2.944
0.04	95.23	133303521	508.59	2.897
0.12	94.04	123057783	469.50	2.527
0.13	89.28	121588415	463.90	2.486
0.14	76.18	120345493	459.16	2.437

Fig. 4. Readings of Multi-Layer pruning with L2-regularisczation channel selection strategy.

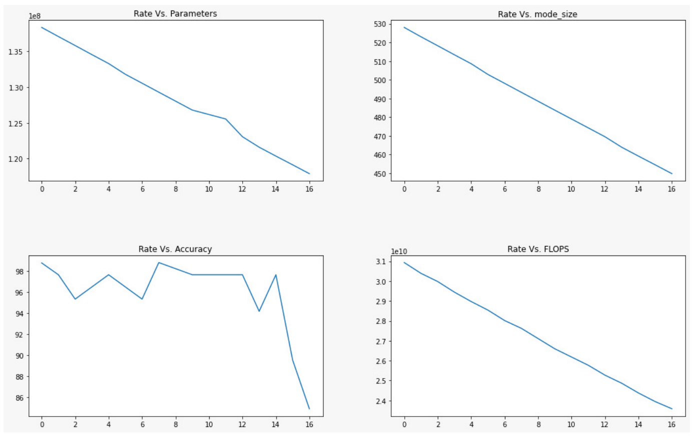


Fig. 5. Performance visualization of Multi-Layer pruning with L1-regularisczation channel selection strategy.

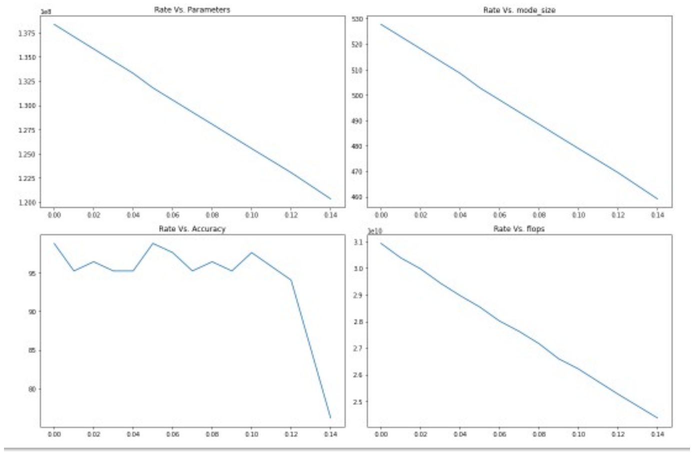
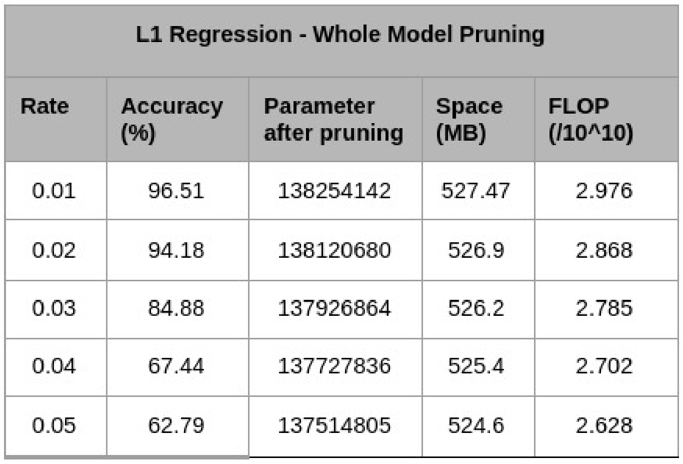


Fig. 6. Performance visualization of Multi-Layer pruning with L2-regularisczation channel selection strategy.

Fig. 5 and Fig 6 here shows plot of evaluation metrics for multi layer pruning in case of L1 and L2 regression. All other metrics other than accuracy decreases with increase in rate. While pruning, it is observed that accuracy might initially increase by small amount but will eventually start decreasing as rate of pruning increases.

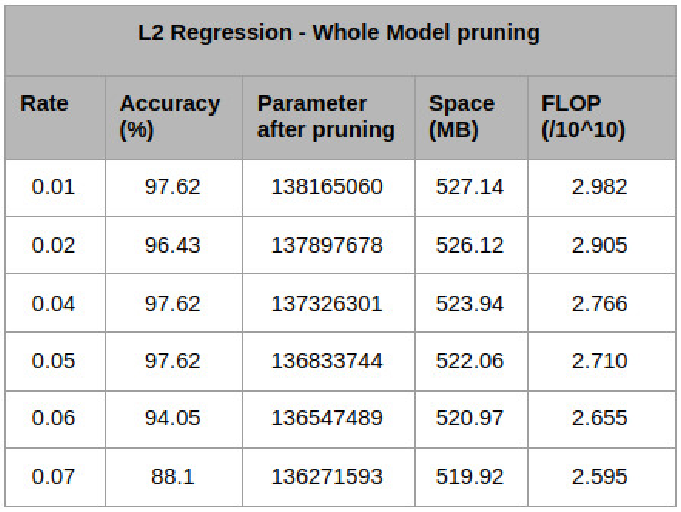
B. Whole Model Pruning

In this sub-section, we focused on how our models performed under whole model pruning. In this case, we are given a pruning rate 'r' which suggests we need to select r% of the least representative over the whole model. Results for both of these methods are given in tables below in fig 7. and fig. 8.



L1 Regression - Whole Model Pruning				
Rate	Accuracy (%)	Parameter after pruning	Space (MB)	FLOP (/10^10)
0.01	96.51	138254142	527.47	2.976
0.02	94.18	138120680	526.9	2.868
0.03	84.88	137926864	526.2	2.785
0.04	67.44	137727836	525.4	2.702
0.05	62.79	137514805	524.6	2.628

Fig. 7. Readings of whole model pruning with L1-regulariszation channel selection strategy.



L2 Regression - Whole Model pruning				
Rate	Accuracy (%)	Parameter after pruning	Space (MB)	FLOP (/10^10)
0.01	97.62	138165060	527.14	2.982
0.02	96.43	137897678	526.12	2.905
0.04	97.62	137326301	523.94	2.766
0.05	97.62	136833744	522.06	2.710
0.06	94.05	136547489	520.97	2.655
0.07	88.1	136271593	519.92	2.595

Fig. 8. Readings of whole model pruning with L2-regulariszation channel selection strategy.

In case of L1 regression, we achieved the best performance in case of pruning rate of 2% with 8% reduction in floating point operations. In case of L2 regression we could achieve 15% reduction in FLOP with pruning rate of 6%.

Fig. 9 and Fig 10 below shows plot of behavior of different parameter for evaluation

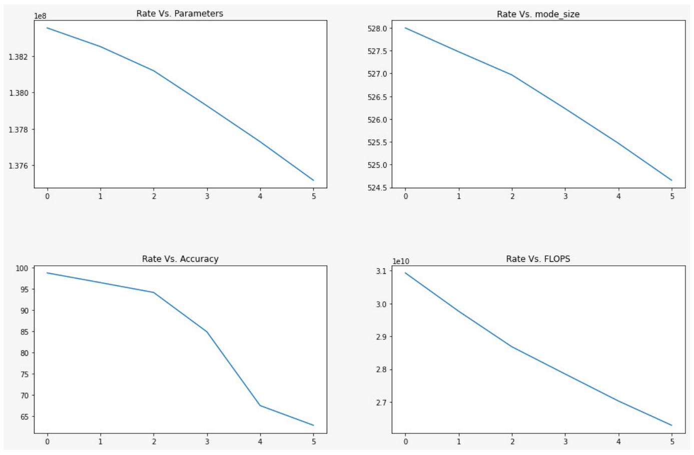


Fig. 9. Performance visualization of whole model pruning with L1-regulariszation channel selection strategy.

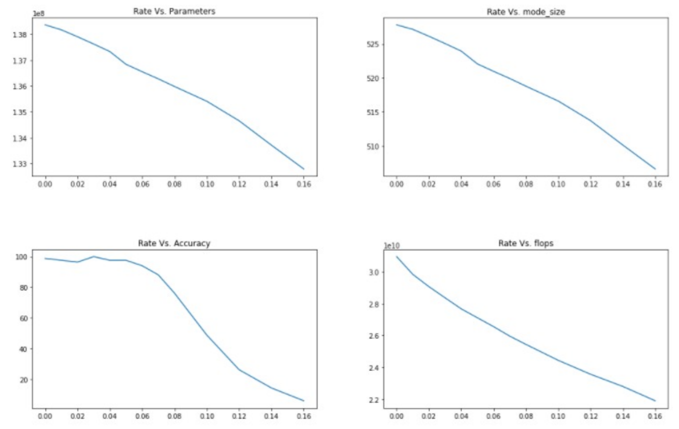


Fig. 10. Performance visualization of whole model pruning with L2-regulariszation channel selection strategy.

VII. Conclusion and Future Work

We have performed comparative analysis to find the best method for channel pruning. Here, we considered L1 and L2 regularization method for channel selection for multi layer and whole model pruning and observed that multi layer pruning performed better than whole model pruning. In multi-layer pruning, L1 method of channel selection outperformed L2 one. While in whole model pruning, L2 method outperformed L1 one method of channel selection. Best result that we could find was in case of multi layer pruning with L1 regression for channel selection and 22% reduction in floating point operation and 14% reduction in parameters.

Channel pruning is all about better selection of channels and future work may include Focusing on other approaches that can be used for channel selection. Moreover, things learned from channel pruning of VGG-16 mode can be then extended to channel pruning of other state-of-the-art models like ResNet, DenseNet, etc.

References

[1] Hengyuan Hu - Rui Peng - Yu-Wing Tai - Chi-Keung Tang [*Network Trimming: A Data-Driven Neuron PruningApproach towards Efficient Deep Architectures*].
`https://arxiv.org/pdf/1607.03250.pdf`

[2] Pravendra Singh, Vinay Kumar Verma, Piyush Raiand, Vinay P. Namboodiri [*Adaptive Filter Pruning for Deep Model Compression*].
`https://www.ijcai.org/Proceedings/2019/0480.pdf`

[3] Zhuang LiuJianguo Li- Zhiqiang Shen - Gao Huang - Shoumeng Yan - Changshui Zhang [*Learning Efficient Convolutional Networks through Network Slimming*].
`https://arxiv.org/pdf/1708.06519.pdf`

[4] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, Guiguang Ding [*Lossless CNN Channel Pruning via Decoupling Remembering and Forgetting*].
`https://arxiv.org/abs/2007.03260`

[5] Anuj Shah [*Model Pruning in Keras with Keras-Surgeon*].
`https://medium.com/exploring-neurons/model-pruning-in-keras-with-keras-surgeon-e8e6ff439c07`

[6] Abhishek Kumar, Nitin Sharma [*Code of this project*].
`https://github.com/weasel-codes/channel-pruning`

[7] [*Kerop Library*].
`https://github.com/kentaroy47/keras-Opcounter`

[8] [*Keras Surgeon*].
`https://github.com/BenWhetton/keras-surgeon`

[9] Yihui He - Xiangyu Zhang - Jian Sun [*Channel Pruning for Accelerating Very Deep Neural Networks*].
`https://ieeexplore.ieee.org/document/8237417`

[10] Zhengtao Wang - Ce Zhu - Zhiqiang Xia - Qi Guo - Yipeng Liu [*Towards Thinner Convolutional Neural Networks Through Gradually Global Pruning*].
`https://ieeexplore.ieee.org/document/8237417`

[11] J. M. Alvarez - M. Salzmann. [*Learning the number of neurons in deep networks.*].
`https://arxiv.org/abs/1611.06321`

[12] Sajid Anwar- Kyuyeon Hwang - Wonyong Sung [*Structured Pruning of Deep Convolutional Neural Networks*].
`https://arxiv.org/abs/1512.08571`

[13] Sajid Anwar - Wonyong Sung [*Structured Pruning of Deep Convolutional Neural Networks*].
`https://arxiv.org/abs/1610.09639`