Similarity-aware Channel Pruning

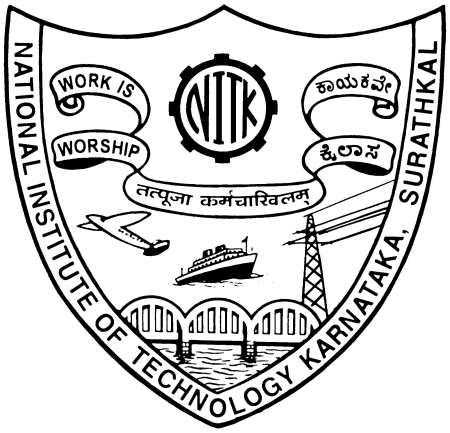
Report of IT499 Major Project-II
Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
INFORMATION TECHNOLOGY
by

Bhagyashri Bhamare(181IT111)
Gagandeep KN (181IT215)

under the guidance of

Dr. Biju R Mohan



DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575025

April, 2022

DECLARATION

We hereby *declare* that the Project Work Report entitled "Similarity-aware Channel Pruning", which is being submitted to the **National Institute of Technology Karnataka, Surathkal**, in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology in the Department of Information Technology, is a *bonafide report of the work carried out by us.*The material contained in this Report has not been submitted at any University or Institution for the award of any degree.

Registration Number, Name & Signature of the Student(s)

(1) 181IT111 - Bhagyashri Bhamare -

(2) 181IT215 - Gagandeep KN -

Department of Information Technology
Place: NITK,Surathkal
Date: 07/04/2022

CERTIFICATE

This is to *certify* that the Project Work Report entitled "Similarity-aware Channel Pruning", submitted by

Sl. No., Registration Number & Name of the Student(s)

(1) 181IT111 - Bhagyashri Bhamare

(2) 181IT215 - Gagandeep KN

as the record of the work carried out by them, is *accepted as the B.Tech. Project Work report submission* in partial fulfillment of the requirement for the award of degree of **Bachelor of Technology** in Information Technology in the Department of Information Technology.

Guide
Dr. Biju R. Mohan

Dr. Jaidhar C. D
Chairman - DUGC

ABSTRACT

In many image identification tasks, deep convolutional neural networks (CNNs) have lately demonstrated substantial accuracy gains. Existing deep convolutional neural network models, on the other hand, are computationally and memory costly, preventing their use on devices with limited memory or in applications with severe latency requirements. As a result, doing model compression and acceleration in deep CNNs without considerably lowering classification accuracy is an obvious concept. In the last few years, huge strides have been made in this field.

We use a new channel pruning approach called Similarity aware Channel Pruning to speed and compress CNNs simultaneously in this research. The majority of known channel pruning techniques are based on filter saliency. However, a modest size does not always imply allow importance of the outputs. As a result, we first compare the similarity of a layer's output feature maps to identify the pruning channels. We locate all the corresponding weight groups (filters, mean, variance, bias,etc.) and select which channels may be eliminated based on the similarity.Then, using the Weight Combination method, we generate new parameters for the following layer in order to rebuild the outputs with fewer channels. The Weight Combination approach also makes fine-tuning and recovering accuracy simpler. After experimenting pruning using different similarity metrics, cosine and euclidean performed good in terms of pruning percentage and accuracy. chebyshev distance based model produces feasible outputs after similarity threshold of 0.80.

.

Keywords— Channel Pruning, Convolutional Neural Networks, Closeness Evaluation, Weight Combination

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 OVERVIEW

A **Convolutional Neural Network** (ConvNet/CNN) is a Deep Learning system that can take an input picture, assign relevance (learnable weights and biases) to various objects in the image, and distinguish between them. When compared to other classification methods, the amount of preprocessing required by a ConvNet is significantly less. While basic approaches need hand-engineering of filters, ConvNets can learn these characteristics with enough training. Deep convolutional neural networks are difficult to deploy on resource-constrained devices because of their high computation and memory requirements. We can tackle this obstacle by compressing the network by pruning it.

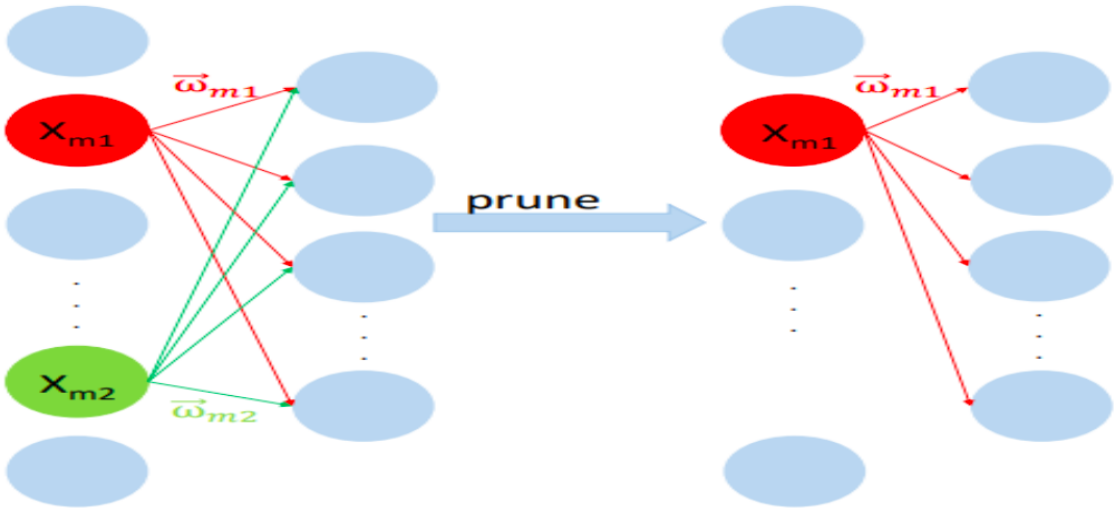


Figure 1.1.1: Pruning example.

Neural network pruning is a method of compression that involves removing weights from a trained model. Pruning a neural network may be done in a variety of ways. Weights can be pruned. Setting individual parameters to zero and making the network sparse accomplishes this. The number of parameters in the model would be reduced while the architecture remained same. You may also take out whole nodes

from the network. This would reduce the size of the network design while maintaining the accuracy of the bigger network.There are different types of pruning, for example Weight-level pruning, Channel Pruning, Filter-level pruning etc

Channel Pruning is the process of eliminating weights from the network that connect neurons from two neighbouring levels. When a DL model contains a large number of convolutional layers, the process of obtaining a near-optimal solution with a stated and acceptable reduction in accuracy becomes more complex. After performing channel pruning, the outputs of the next layer should be reconstructed by combining the corresponding channel-wise weights of all filters of the next layer.That means when two feature maps are identified strong relative similarity, one of them can be removed safely, the related channel-wise weights will be discarded too. By applying such an addition operation to the corresponding channel-wise weights, can get a reconstructed feature map that is not much different from the original feature map. This process is known as **Weight Reconstruction**

In machine learning, **transfer learning** refers to the reuse of a previously learned model on a new issue.In transfer learning, we first train a base network on a base dataset and task, , or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.In this project we are using transfer learning to train a base network and then we repurpose the learned features.

The effect of applying the filters to an input image is captured by the **feature maps** of a CNN. The **feature map** is the output of each layer, i.e. at each layer, the feature map is the output of that layer. The goal of visualising a feature map for a given input picture is to learn more about the features that our CNN finds.

We implement a Similarity-aware Channel Pruning technique in this project.By utilising the similarity of a layer's output feature maps, we aim to eliminate duplicated channels. Then, using the Weight Combination technique, we produce new parameters for the following layer in order to rebuild the outputs. Our project's main points may be described as follows:

- Instead of concentrating on the filter saliency, we do channel pruning by focusing on the redundancy of feature maps directly.

- A Similarity-aware Channel Pruning technique will be utilised to minimise repetition by rejecting the related channels of the next layer that use the deleted feature maps as inputs and deleting the unnecessary weight groups of the current layer that create similar outputs.

- We will use the Weight Combination approach to generate new parameters for the following layer, allowing us to reconstruct the outputs with fewer channels. Through fine-tuning, this technique makes it simpler to restore precision.

- Conduct experiments on CIFAR-10 to determine the efficiency Most of the existing

channel pruning approaches mainly prune unimportant filters by the filter saliency. However, filters with low saliency doesn't always lead to low saliency to the outputs. Under such circumstance, the incorrect removal may result in the accuracy reduction of the model. Moreover, a direct method of channel pruning is to exploit the filter similarity and remove the redundant ones which have the similar values to others. However, such kind of methods have some limitations:

- Influenced by other factors (such as bias, batch normalization), similar filters don't necessarily lead to similar feature maps. Therefore, the removal of the similar filters may result in the incorrect pruning.

- Different filters can also generate similar feature maps. Thus, merely comparing the similarity of the filters may lead to ignoring the removal of redundant filters

1.2 MOTIVATION

Our project is motivated by two factors. First, despite Convolutional Neural Networks' (CNNs) tremendous performance in different visual identification tasks, their high computational and storage costs prevent them from being used in resource-constrained devices. To get around this, we'll utilise channel pruning to thin down the network. Second, current techniques perform channel pruning by reducing the feature map reconstruction error between the pretrained and pruned models. Although reconstructing feature maps can maintain the majority of the information in the learnt

model, it has drawbacks. Deep Convolutional Neural Networks (CNNs) have a large computational power and memory footprint, which prevents them from being used on edge computing devices like smart phones or wearable electronics. Extensive research has been done on compressing CNNs to alleviate this issue. Channel pruning has been identified as one of the most effective strategies for compressing CNNs among them. Channel pruning aims to remove the complete channel from each layer, which is simple yet difficult since eliminating channels from one layer might radically alter the input of the following layer.

Chapter 2

LITERATURE REVIEW

2.1 BACKGROUND AND RELATED WORKS

H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf present an acceleration approach for CNNs in the work "Pruning filters for efficient convnets", in which they remove filters from CNNs that are found to have a minimal influence on output accuracy. The calculation costs are considerably lowered by eliminating whole filters from the network together with their connected feature maps. This method, unlike pruning weights, does not result in sparse connection patterns.On CIFAR10, they show that even basic filter pruning approaches may cut inference costs for VGG-16 by up to 34the networks to near-original accuracy.

In the work "Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks" by **He, G. Kang, X. Dong, Y. Fu, and Y. Yang**, the Soft Filter Pruning (SFP) approach was introduced to speed up the inference phase of deep Convolutional Neural Networks (CNNs). The suggested SFP, in particular, allows the pruned filters to be updated during model training following pruning. In comparison to earlier works, SFP offers two advantages: (1) A model with a larger capacity. When we update previously pruned filters, we have more optimization space than when we fix the filters to zero. As a result, the network trained using our technique has a greater potential to learn from training data. (2) There is a lower reliance on the pre-trained model. SFP can train from scratch and prune the model at the same time because of its large capacity. 3

In the Paper "Pruning convolutional neural networks for resource efficient inference", **P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz** offer a novel approach for pruning convolutional kernels in neural networks to allow for more efficient inference. They combine greedy criteria-based pruning with back propagation fine-tuning, a computationally efficient method that keeps the trimmed network's generalisation. They offer a novel Taylor expansion-based criteria for estimating the change in the cost function caused by trimming network parameters. They concentrate on transfer learning, which involves adapting vast, pre-trained networks to

specific tasks. Finally, they demonstrate the versatility of our method by displaying findings from the large-scale ImageNet dataset.

Zi Wang, Chengcheng Li offer a systematic channel pruning approach in the paper "Towards Efficient Convolutional Neural Networks through Low-Error Filter Saliency Estimation" that considerably decreases the filter saliency estimate error. By including the alternate direction method of multipliers (ADMM) into the pre-training phase, our method significantly decreases the magnitude of parameters in a network. As a result, the Taylor expansion-based estimation of filter saliency has considerably improved. This article conducts extensive tests with a variety of benchmark networks.

Kaveena Persand, Andrew Anderson, David Gregg offer a new classification of saliency measures based on four major components that are generally orthogonal in the paper "Taxonomy of Saliency Metrics for Channel Pruning". They show that these components may be used to aggregate a wide range of measures from the pruning literature. Their classification guides previous work and helps us to create new saliency measurements by experimenting with different combinations of their taxonomic components. They conduct the first in-depth experimental examination of more than 300 saliency measures composed of known approaches and new component combinations. These findings give definitive solutions to research issues that have remained unanswered. They illustrate the relevance of reduction and scaling when trimming weight groups in particular.

Yihui He; Xiangyu Zhang; Jian Sun provide a new channel pruning approach for accelerating very deep convolutional neural networks in the paper "Channel Pruning for Accelerating Very Deep Neural Networks". They offer an iterative two-step method that uses a LASSO regression-based channel selection and least square reconstruction to successfully prune each layer given a trained CNN model. We extend our approach to multi-layer and multi-branch situations. This technique lowers the total error and improves compatibility with a variety of architectures.

Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, Jin-Hui Zhu examine discrimination-aware channel pruning (DCP), a simple-yet-effective approach for selecting channels that truly contribute to discriminative power, in the work "Discrimination-aware Channel Pruning for Deep Neural Networks". To do this, they insert extra discrimination-aware

losses into the network to boost the discriminative power of intermediate layers, and then use the additional loss and reconstruction error to pick the most discriminative channels for each layer. Finally, they offer an iterative channel selection and parameter optimization technique based on a greedy algorithm. Extensive testing has shown that this approach is successful.

Yu Cheng, Duo Wang, Pan Zhou, Tao Zhang review the most current advanced approaches for compacting and speeding CNN models described in the paper "A Survey of Model Compression and Acceleration for Deep". Parameter pruning and sharing, low-rank factorization, transferred/compact convolutional filters, and knowledge distillation are the four schemes that these approaches fall within. The methods of parameter trimming and sharing are discussed first, followed by the introduction of the other approaches. They give in-depth study of each scheme's performance, associated applications, benefits, and disadvantages, among other things.

Jose M Alvarez, Mathieu Salzmann present a method for automatically calculating the number of neurons in each layer of a deep network during learning in the study "Learning the Number of Neurons in Deep Networks". To do this, they propose using a group sparsity regularizer on the network's parameters, with each group specified to act on a single neuron.

Kwangbae Lee; Hoseung Kim; Hayun Lee; Dongkun Shin focus on the group-level pruning approach to speed deep neural networks on mobile GPUs in the study "Flexible Group-Level Pruning of Deep Neural Networks for On-Device Machine Learning", which involves pruning multiple neighbouring weights in a group to reduce the irregularity of pruned networks while maintaining high accuracy. Despite the fact that numerous group-level pruning strategies have been presented, the prior algorithms chose weight groups to be trimmed at places that were group-size aligned. They suggest a more flexible method termed unaligned

2.2 OUTCOME OF LITERATURE REVIEW

.

Table 2.2.1: Literature survey Table

Author	Title	Findings	Advantage	Gap in literature
P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz	Pruning convolutional neural networks for resource efficient inference	In this Paper, they offer a novel approach for pruning convolutional kernels in neural networks to allow for more efficient inference.	adapting vast, pre-trained networks to specific tasks	greedy criteria-based pruning is used
H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf	Pruning filters for efficient convnets	They present an acceleration approach for CNNs in this paper, in which they remove filters from CNNs that are found to have a minimal influence on output accuracy	No sparse connection patterns	Only influence on the output accuracy is considered
Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang	Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks	The Soft Filter Pruning (SFP) approach was introduced in this study to speed up the inference phase of deep Convolutional Neural Networks (CNNs)	There is a lower reliance on the pre-trained model	Relation between the filters is not considered
Zi Wang, Chengcheng Li	Towards Efficient Convolutional Neural Networks through Low-Error Filter Saliency Estimation	They offer a systematic channel pruning approach based on alternate direction method of multipliers (ADMM)	Considerably decreases the filter saliency estimate error	Includes only filter saliency as a major factor

S.no	Author	Title	Findings	Gap in literature
Kaveena Persand, Andrew Anderson, David Gregg	Taxonomy of Saliency Metrics for Channel Pruning	They offer a new classification of saliency measures based on four major components that are generally orthogonal.	in-depth experimental investigation of more than 300 saliency metrics	-
Yihui He; Xiangyu Zhang; Jian Sun	Channel Pruning for Accelerating Very Deep Neural Networks	They offer an iterative two-step method that uses a LASSO regression-based channel selection and least square reconstruction.	Extends approach to multi-layer and multi-branch situations	-
Zhuangwei Zhuang, Mingkui Tan, et. la	Discrimination-aware Channel Pruning for Deep Neural Networks	They examine discrimination-aware channel pruning (DCP), a simple-yet-effective approach for selecting channels that truly contribute to discriminative power,in this work.	Similarity between the filters is considered for selecting channels to prune	-
Yu Cheng, Duo Wang, Pan Zhou, Tao Zhang	A Survey of Model Compression and Acceleration for Deep	They review the most current advanced approaches for compacting and speeding CNN model	Comparative analysis is done	Review paper (No limitations)

- Referred papers provided better understanding of channel pruning, it's existing methods and complexities involved.
- Insight into limitations of existing techniques of channel pruning was obtained
- Advantages and drawbacks of channel pruning was understood
- Comparison between different current advanced approaches for compacting and speeding CNN models was clear by the survey paper
- Method for automatically calculating the number of neurons in each layer of a deep network during learning using group sparsity regularizer was understood

2.3 PROBLEM STATEMENT

Implentation of Similarity-aware Channel Pruning to simultaneously accelerate and compress CNNs to make increase the feasiblility of deploying CNNs on resource-constrained devices.

2.4 RESEARCH OBJECTIVES

- Achieve Deep neural network model compression and acceleration via channel pruning based on an unique Similarity-aware Channel Pruning technique to fasten the computation speed
- Implement different technique to find the similarities between the channel or to find the cluster of similar channels
- Conduct extensive tests on CIFAR-10 to compare the models.

Chapter 3

PROPOSED METHODOLOGY

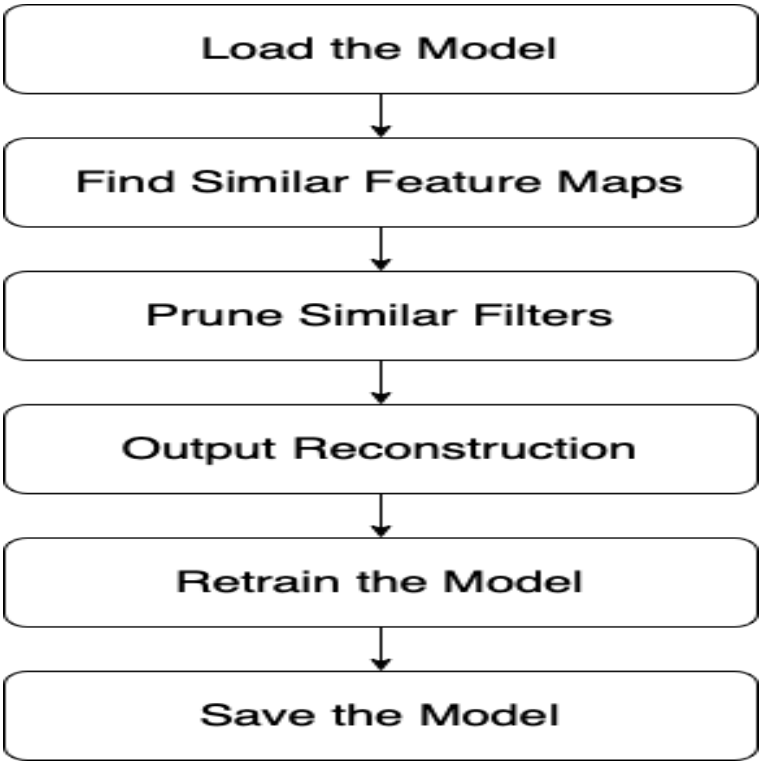


Figure 3.0.1: Work Flow.

In the Fig. 3.0.1, the basic work flow of the project is shown. The model is loaded in the first step to work on the compression of it. TO compress the model or to prune the network, we find the similar feature maps and remove these redundant features. After pruning, output reconsutuction, training and fine tuning is carried out. And these weight values of the network is saved for furture utilisation.

3.1 NOTATIONS

Suppose the weights of convolutional layers in a CNN can be represented as

$$W^{(l)} \in R^{N_l \times C_l \times K_l \times K_l}, 1 \leq l \leq L \tag{3.1}$$

where $W^{(l)}$ denotes a 4D weight tensor with $K_l \times K_l$ kernel size in the l-th layer, N_l and C_l are the dimensions of the weight tensor $W^{(l)}$ along the axes of filter and channel respectively. L denotes the number of convolutional layers. A set of output feature maps of the l$\hat{\text{t}}$h layer can be expressed as

$$Z^{(l)} \in R^{N_l \times H_l \times W_l} \; with \; size \; H_l \times W_l, \tag{3.2}$$

which are used as input feature maps of the next layer as well. Moreover, let ${Z_k^{(l)}}_{,:,:}$ be the k-th channel of $Z^{(l)}$, ${Z_k^{(l)}}_{,:,:}$ is obtained by applying the the convolutional operator $(*)$ to the input feature maps $Z^{(l-1)} \in R^{N_{l-1} \times H_{l-1} \times W_{l-1}}$ with the k-th filter of $W^{(l)}$, represented as ${W_k^{(l)}}_{,:,:,:}$, i.e.,

$${Z_k^{(l)}}_{,:,:} = f(\Sigma_{c_l=1}^{C_l}(Z_{c_l,:,:}^{(l-1)} * {W_k^{(l)}}_{,c}\, l_{,:,:})) \tag{3.3}$$

where f$(\cdot)$ is a non-linear activation function, $W_{k,c_l,:,:}^{(l)}$ is the kernel of the k-th filter and $c_l th$ channel for $W^{(l)}$

3.2 DATASET

In this project, we use CIFAR-10 dataset to train the VGG16 network to later perform pruning operation and retrain the model. CIFAR-10 is an established computer-vision dataset used for object recognition. It is a subset of the 80 million tiny images dataset and consists of 60,000 32x32 color images containing one of 10 object classes, with 6000 images per class. 50,000 images are used for training and 10,000 is used for testing in this project.

In Fig 3.2.1, Example images from the dataset is visualised.

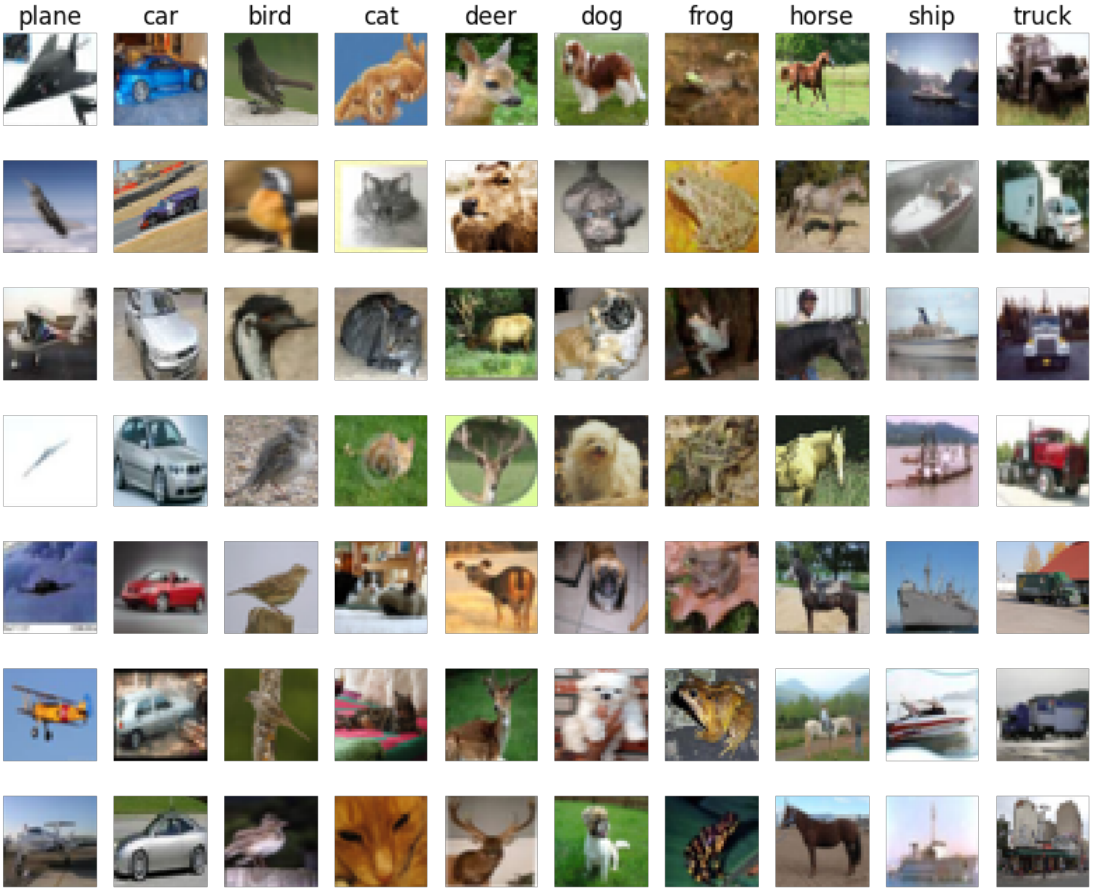


Figure 3.2.1: Sample image shots of CIFAR-10 Dataset

3.3 TRAIN AND LOAD THE MODEL

Model is trained on CIFAR-10 dataset and this model is loaded from the local to perform pruning operations to compress and accerate the model. We use VGG16 in this projectVGG16 is a convolution neural net (CNN) architecture that won the ILSVR(Imagenet) competition in 2014. It is regarded as one of the most outstanding vision model architectures ever created. The most distinctive feature of VGG16 is that, rather than having a huge number of hyper-parameters, they concentrated on having 3x3 filter convolution layers with a stride 1 and always utilised the same padding and maxpool layer of 2x2 filter with stride 2. Throughout the design, it maintains this convolution and max pool layer layout. Finally, it has two FC (completely connected layers) for output, followed by a softmax. The 16 in VGG16 points to the fact that it comprises 16 layers, each of which has a different weight. This network is huge, with around 138 million (estimated) parameters. Deep model compression and acceleration can be achieved via channel pruning.

Figure 3.3.1: Architecture of VGG16 network [13]

3.4 FIND SIMILAR FILTERS

To determine which channels of each layer can be removed safely, we can first compare the similarity of output feature maps. Thus, we should collect a representative dataset to evaluate the similarity between output feature maps, which is called an evaluation set. Moreover, for each layer, we need to define a similarity matrix that introduces the k-nearest neighbors (KNN) method to better measure the relative similarity between two output feature maps.

In this project, we have started implementing four similarity criteria to eliminate the redundant feature map. First one is cosine similarity which measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. $s_{ij}^{(l)}$ can be defined as:

$$s_{ij}^{(l)} = (cosine\, of\, angle\, between(z_i^{(l)}, z_j^{(l)})) \tag{3.4}$$

Euclidean distance between two points in Euclidean space is the length of a line segment between the two points. Euclidean distance is calculated as the square root of the sum of the squared differences between the two vectors.Euclidean distance $d_{ij}^{(l)}$ can be defined as:

$$d_{ij}^{(l)} = \sqrt{\sum_{n=1}^{n=N} (z_i^{(l)}[n] - z_j^{(l)}[n])^2} \tag{3.5}$$

where N is number of dimensions

Manhattan distance, The total of the absolute differences between the two vectors is used to determine the Manhattan distance. The L1 vector norm, as well as the sum absolute error and mean absolute error metrics, are all connected to the Manhattan distance, where Manhattan distance $d_{ij}^{(l)}$ can be defined as:

$$d_{ij}^{(l)} = \sum_{n=1}^{n=N} |z_i^{(l)}[n] - z_j^{(l)}[n]| \tag{3.6}$$

where another similarity criteria is Chebyshev distance, also known as maximum metric or L metric in mathematics. It is a metric defined on a vector space in which the

distance between two vectors is the biggest of their differences along any coordinate dimension. where Chebyshev distance $d_{ij}^{(l)}$ can be defined as:

$$d_{ij}^{(l)} = \max_{n=1}^{n=N} |z_i^{(l)}[n] - z_j^{(l)}[n]| \tag{3.7}$$

3.5 PRUNE SIMILAR FILTERS (CHANNEL PRUNING)

To further determine which of these two similar feature maps should be removed, we define the pruning loss to measure the impact on model performance induced by removing one of them. we can remove the feature map that brings less loss, while preserving the other for keeping the feature information. For example Taking the l-th layer as an example, we define the similarity matrix as $S^{(l)} = (s_{ij}^{(l)}) \in R^{N_l \times N_l}$, where $s_{ij}^{(l)}$ represents the similarity between output feature maps $Z_{i,:,:}^{(l)}$ and $Z_{j,:,:}^{(l)}$ by cosine similarity and . Let $z_i^{(l)}$ and $z_j^{(l)}$ be the vectors expanded in the row of $Z_{i,:,:}^{(l)}$ and $Z_{j,:,:}^{(l)}$ respectively. Therefore, $s_{ij}^{(l)}$ can be defined as:

$$s_{ij}^{(l)} = \cos(z_i^{(l)}, z_j^{(l)}) = \frac{(z_i^{(l)} \cdot z_j^{(l)})}{|z_i^{(l)}| \times |z_j^{(l)}|} .....[0, 1] \tag{3.8}$$

if $z_i^{(l)}$ and $z_j^{(l)}$ are the nearest neighbors then their similarity will be 1. When $z_i^{(l)}$ and $z_j^{(l)}$ aren't the nearest neighbors, $s_{ij}^{(l)}$ is defined as 0.. When the value of $s_{ij}^{(l)}$ is large enough, we can identify the strong relative similarity between ${Z_i^{(l)}}_{,:,:}$ and ${Z_j^{(l)}}_{,:,:}$.
The length of a line segment connecting two locations in Euclidean space is the Euclidean distance between them. Using euclidean distance $s_{ij}^{(l)}$ can be defined as:

$$s_{ij}^{(l)} = \frac{1}{1 + euclidean\ distance((z_i^{(l)}, z_j^{(l)}))} .....(0, 1]. \tag{3.9}$$

if $z_i^{(l)}$ and $z_j^{(l)}$ are the nearest neighbors then their similarity will be 1. When $z_i^{(l)}$ and $z_j^{(l)}$ aren't the nearest neighbors, $s_{ij}^{(l)}$ tend to 0.
The Manhattan distance is a measure for measuring the distance between two points in an N-dimensional vector space. It is the total of the lengths of the line segment projections into the coordinate axes between the points. It is the total of the abso-

lute differences between the measurements in all dimensions of two points in basic words.Using manhattandistance distance $s_{ij}^{(l)}$ can be defined as:

$$s_{ij}^{(l)} = \frac{1}{1 + manhattan\ distance((z_i^{(l)}, z_j^{(l)}))} .....(0, 1]. \tag{3.10}$$

if $z_i^{(l)}$ and $z_j^{(l)}$ are the nearest neighbors then their similarity will be 1. When $z_i^{(l)}$ and $z_j^{(l)}$ aren't the nearest neighbors, $s_{ij}^{(l)}$ tend to 0.

In mathematics, the Chebyshev distance computation, often known as the "maximum metric," calculates the distance between two locations as the largest difference between any of their axis values. In a 2D grid, for example, the Chebyshev distance between two points (x1, y1) and (x2, y2) is maximum (y2 - y1, x2 - x1).Using Chebyshev distance $s_{ij}^{(l)}$ can be defined as:

$$s_{ij}^{(l)} = \frac{1}{1 + Chebyshev\ distance((z_i^{(l)}, z_j^{(l)}))} .....(0, 1] \tag{3.11}$$

if $z_i^{(l)}$ and $z_j^{(l)}$ are the nearest neighbors then their similarity will be 1. When $z_i^{(l)}$ and $z_j^{(l)}$ aren't the nearest neighbors, $s_{ij}^{(l)}$ tend to 0.

To further determine which of these two similar feature maps should be removed, we define the pruning loss $L_p^{(l+1)}$ to measure the impact on model performance induced by removing one of them:

$$L_p^{(l+1)} = \frac{1}{\frac{1}{N_{l+1}} \Sigma_{k=1}^{N_l+1} |O_{k,:,:}^{(l+1)} - Z_{k,:,:}^{(l+1)}|_F^2} \tag{3.12}$$

where $O_{k,:,:}^{(l+1)}$ denotes the original feature map and $Z_{k,:,:}^{(l+1)}$ denotes the compressed one.

By comparing the values $L_p^{(l+1)}$ after discarding $Z_{i,:,:}^{(l)}$ and $Z_{j,:,:}^{(l)}$ respectively, we can remove the feature map that brings less loss, while preserving the other for keeping the feature information.

3.6 OUTPUT RECONSTRUCTION

After removing similar feature maps and related redundant weight groups, the outputs of the next layer can be reconstructed by combining the corresponding channel-wise weights of all filters of the next layer.That means when two feature maps are identified strong relative similarity, one of them can be removed safely, the related channel-wise weights will be discarded too. By applying such an addition operation to the corresponding channel-wise weights, can get a reconstructed feature map that is not much different from the original feature map. Extending the implementation procedure to all output feature maps it's obvious that we can reconstruct all outputs of the next layer accurately by combining the corresponding channel-wise weights of all filters of the next layer.

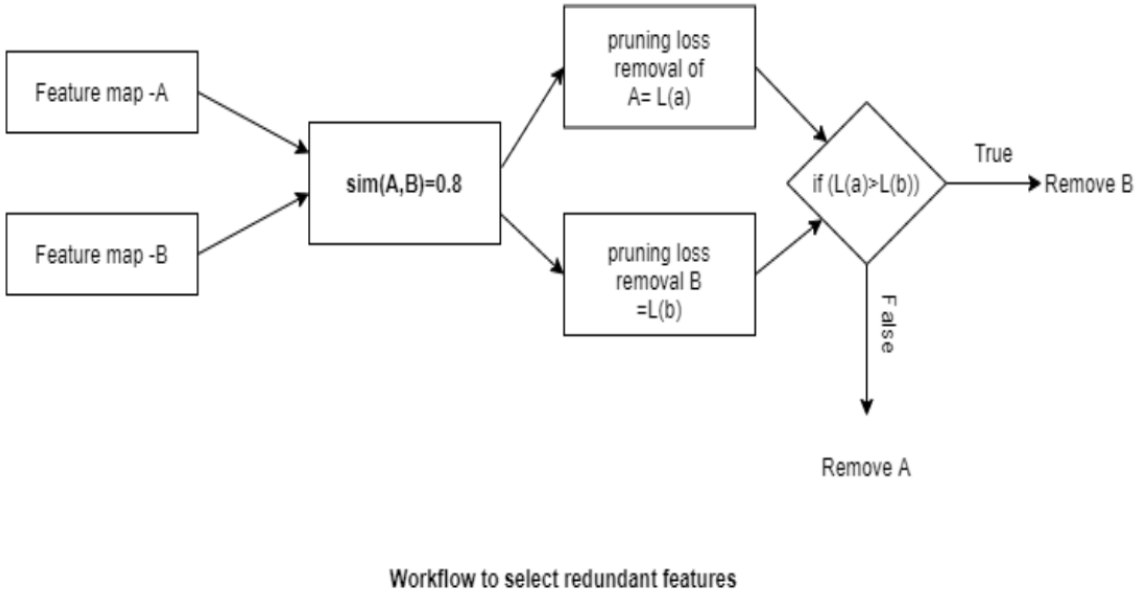


Figure 3.6.1: Overflow to select similar filter A and B

In the example shown in Fig. 3.6.1, the similarity between feature map A and Feature map B is 0.8, which is more than similarity Threshold. Then the feature map that has to be reduced is chosen based on the minimum pruning loss. Therefore if L(A) is greater than L(B), where L denotes the pruning loss, Feature Map A is declared redundant. If the case is otherwise, Feature Map B is declared redundant.

3.7 FLOATING POINT OPERATIONS

To measure inference time for a model, we can calculate the total number of computations the model will have to perform.This is where we mention the term FLOP, or Floating Point Operation.This could be an addition, subtraction, division, multiplication, or any other operation that involves a floating point value. The FLOPs will give us the complexity of our model.Our aim will be to optimize the Deep Learning models to have a low number of FLOPs
To calculate the FLOPs in a model, here are the rules:

$$Convolutions\ FLOPs = 2 \times N_i \times N_{i+1} \times K_i \times K_j \times O_i \times O_j \tag{3.13}$$

where $N_i \times N_{i+1}$ is of Kernel, $K_i \times K_j$ is Kernel Shape and $O_i \times O_j$ is Output Shape

$$Fully\ Connected\ Layers\ FLOPs = 2 \times InputSize \times OutputSize \tag{3.14}$$

$$Total\ no\ FLOPs = Convolutions + Fully\ Connected\ Layers \tag{3.15}$$

3.8 WHOLE MODEL PRUNING AND RETRAINING (RETRAIN THE MODEL)

After performing Channel Pruning and Weight Combination in a layer-by-layer manner, we should retrain the network to further compensate the performance degradation. Specifically, We first use an evaluation set to get output feature maps of all weighted layers and compare the similarity between output feature maps for each layer. Then we prune the redundant channel and generate the new parameters by Weight Combination strategy iteratively for all layers to get a compressed model. Finally we retrain the compressed model to recover the model accuracy.

3.9 FINE TUNE AND SAVE THE MODEL

Fine-tuning is the process in which parameters of a model must be adjusted very precisely in order to fit with certain observations. After fine-tunning the retrained model is obtained and it is saved as the data collection of weights and parameters.

Chapter 4

RESULTS AND ANALYSIS

4.1 EXPERIMENTAL SETUP

All experiments were performed on Intel(r) Core(TM) i7-6700 CPU @ 3.40Ghz and a 8GB ofRAM running a 64-bit Ubuntu 14.04 edition. The software implementation has been in Pytorch library 1 on two Titan X 12GB GPUsThe network pruning was implemented in Pytorch deep learning library (Paszke et al., 2017). We evaluated the proposed redundant-feature-based pruning on three deep networks, namely: VGG-16 (Simonyan and Zisserman, 2015) and all the networks in our experiments are fine-tuned using SGD with a weight decay of 5^{-4} and a Nesterov momentum of 0.9. And the Similarity Threshold S for all weighted layers is set different. The dataset is split into 50000 and 10000 training and testing sets, respectively. We used FLOP to compare the computational efficiencyof the models considered because its evaluation is independent of the underlying software and hardware. In order to fairly compare our method , we also calculated the FLOP only for the convolution and fully connected layers.

4.2 RESULTS

Our base model VGG16 was trained for 350 epochs, with a batch-size of 128 and a learning rate 0.1.We have pruned and retrained VGG16 model which originally had an Accuracy of 93.87% before pruning and later we have compared the accuracies after pruning. After pruning we retrain the network with learning rate of 0.001 for 80 epochs to fine tune the weights of the remaining connections to regain the accuracy.The the graphs plotted for different threshold with different similarity metrics and Accuracy table are as below.

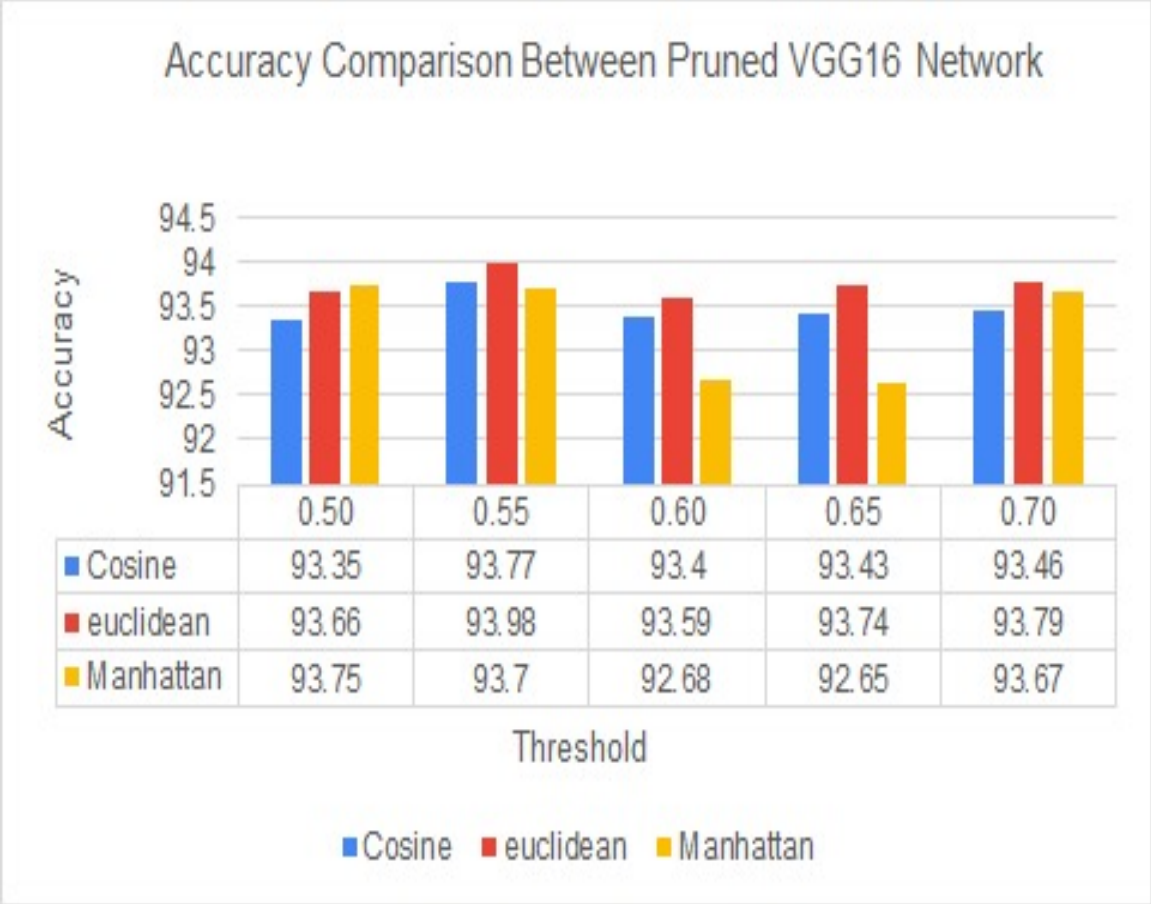


Figure 4.2.1: (Threshold vs Accuracy) Accuracy comparison between pruned network using different similarity criteria

In the Fig. 4.2.1, Maximum accuracy of 93.78% is achieved at threshold 0.55 for the pruned network using euclidean similarity as the distance criteria. Accuracy is higher when euclidean distance is used as similarity criteria, therefore the overall performance of is higher when euclidean distance is used to find the similarity to prune but the reduction in flops, i.e the pruning percentage is less compared to the pruned network where cosine is used.

There is sudden fall and gradual rise observed after threshold = 0.55 in the models pruned using different similarity criteria. The maximum accuracy is achieved for every model at threshold = 0.55.

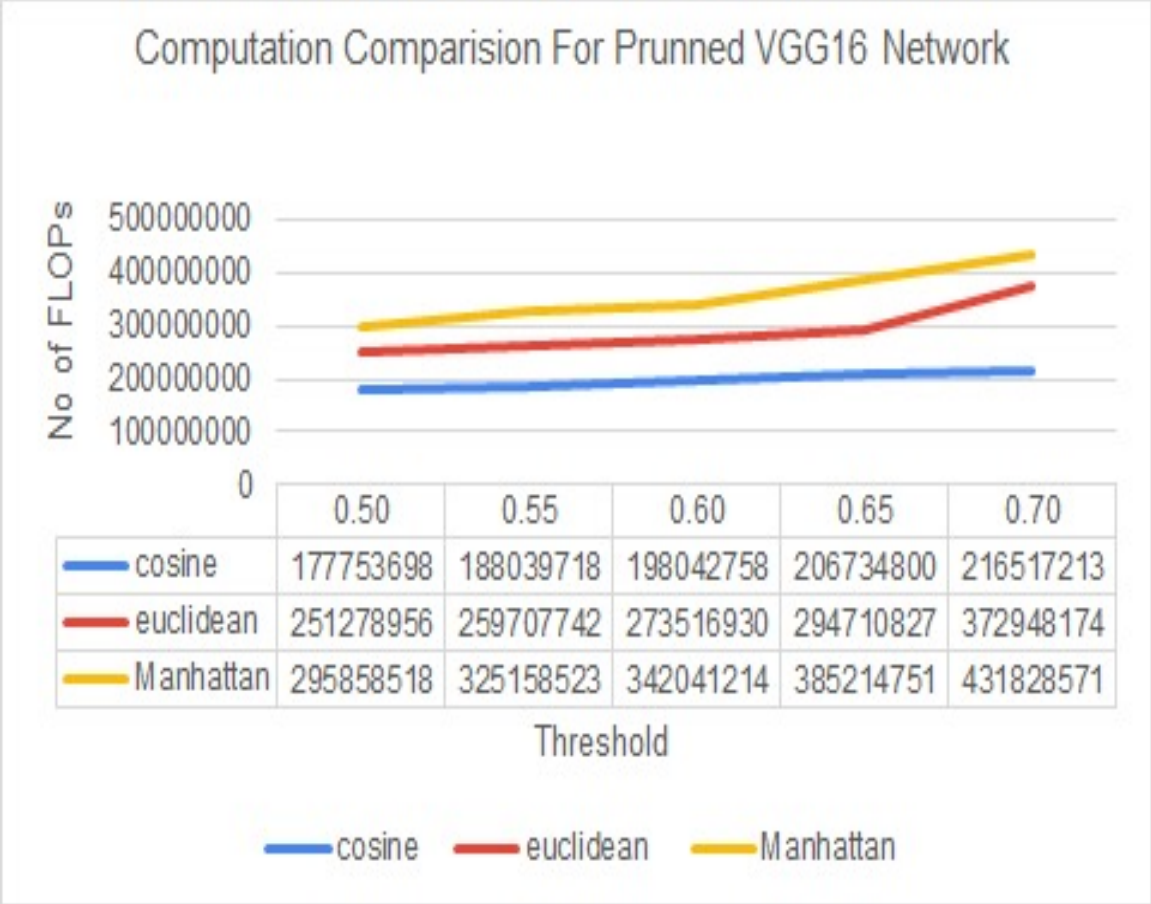


Figure 4.2.2: (Threshold vs FLOPs) comparison between pruned network using different similarity criteria

In the Fig. 4.2.2, minimum number of FLOPs is achieved using Chebyshev distance similarity criteria. Euclidean distance performed better accuracy wise, but the FLOPs are more for euclidean distance, thus taking more time than the pruned model using cosine similarity and Chebyshev distance similarity criteria. When it comes to faster network, model pruned using Chebyshev distance similarity criteria performs better than all the models with every similarity threshold. But the accuracy for Chebyshev is very low to use the model in practical applications. Second best model computation wise will be cosine similarity as the accuracy of this model is decent and the FLOPs are less than Euclidean distance, thus running faster.

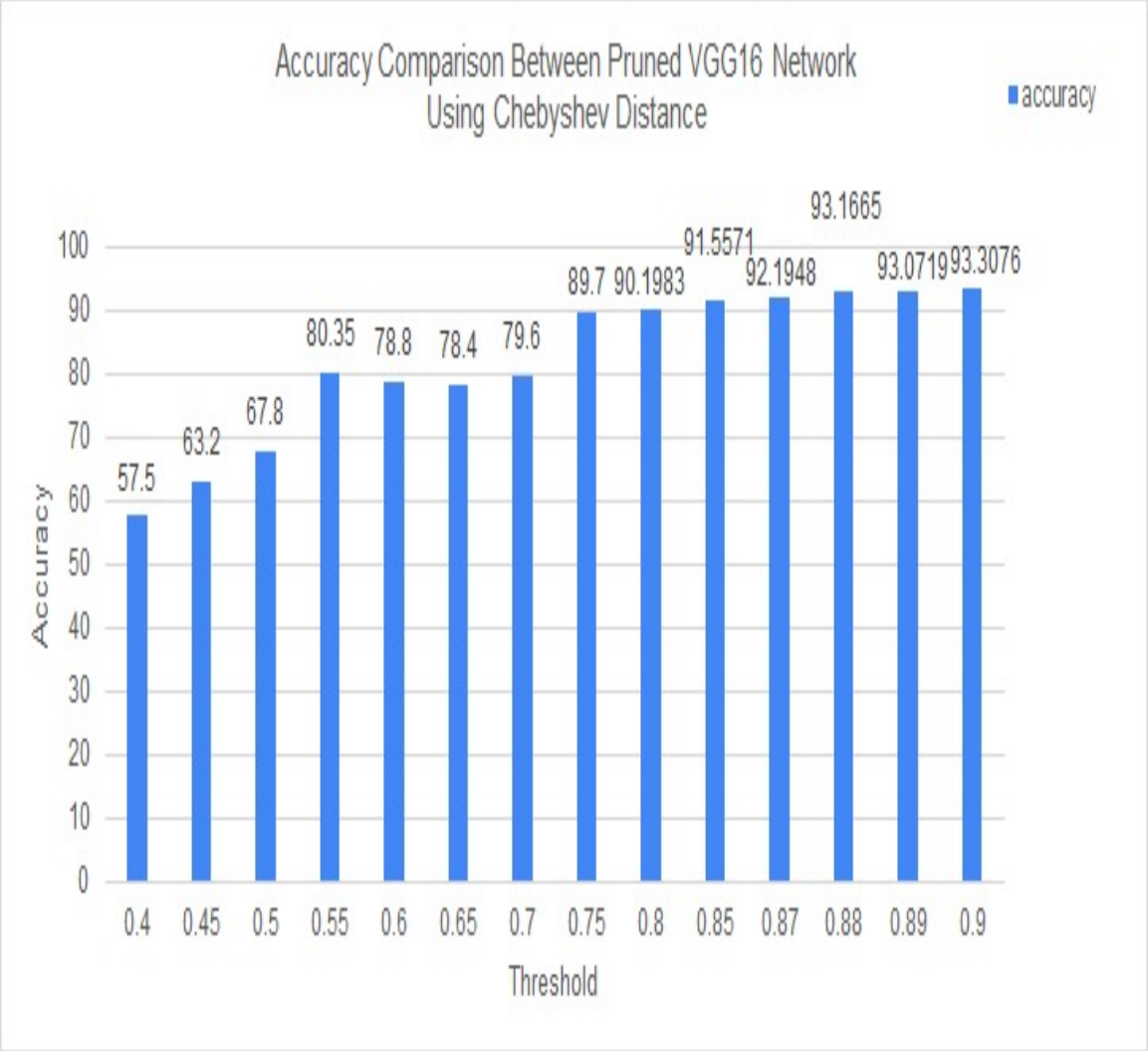
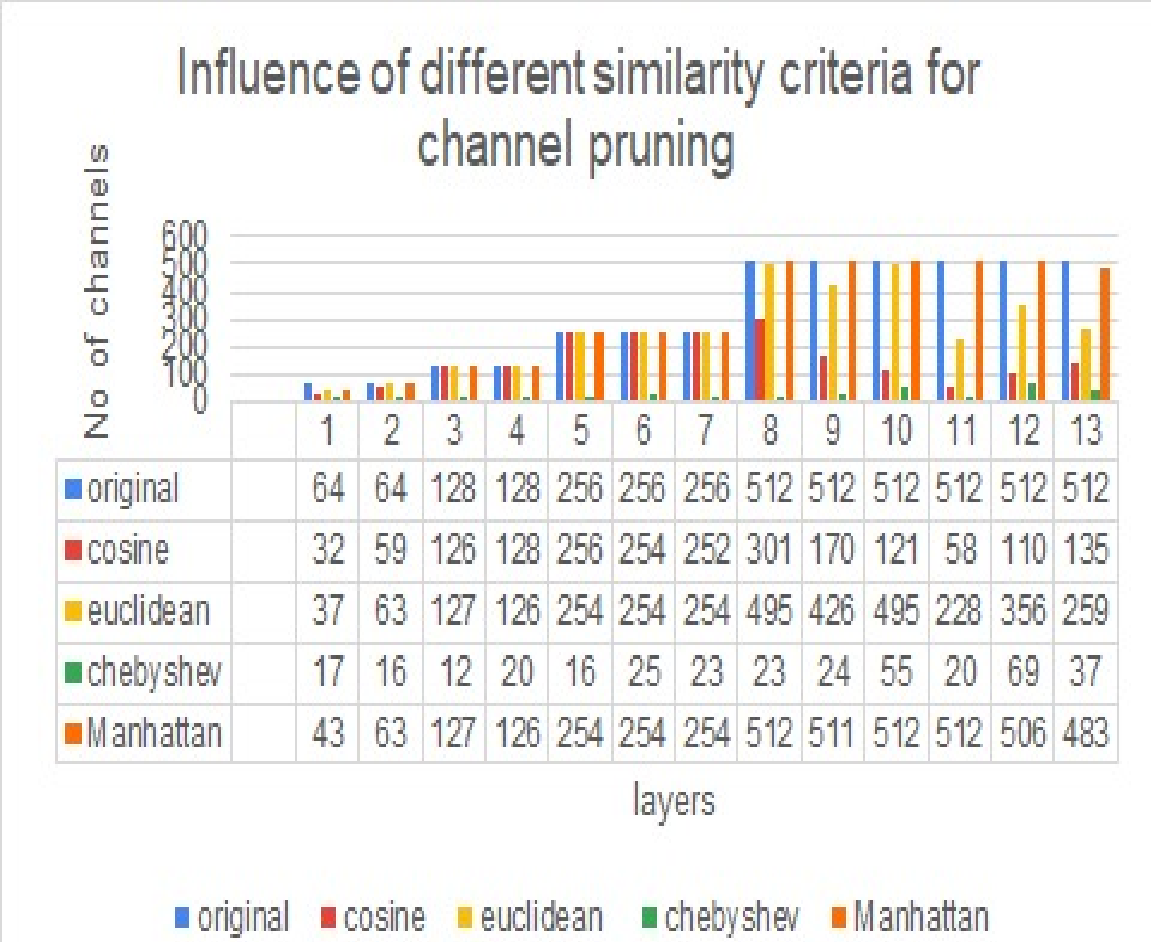


Figure 4.2.3: (Threshold vs accuracy) comparison between pruned network using chebyshev distance similarity criteria along the range of threshold.

In Fig 4.2.3, we can observe that chebyshev is feasible, if higher accuracy is a requirement, only from threshold similarity of 0.85 to 0.9. Even with 0.90 as threshold, the model is pruned to greater extent, producing very little loss of accuracy at this threshold.



	1	2	3	4	5	6	7	8	9	10	11	12	13
original	64	64	128	128	256	256	256	512	512	512	512	512	512
cosine	32	59	126	128	256	254	252	301	170	121	58	110	135
euclidean	37	63	127	126	254	254	254	495	426	495	228	356	259
chebyshev	17	16	12	20	16	25	23	23	24	55	20	69	37
Manhattan	43	63	127	126	254	254	254	512	511	512	512	506	483

Figure 4.2.4: (Layers vs No of channels at threshold = 0.55) Layer wise comparison between pruned network using different similarity criteria with original network

In the Fig. 4.2.4, Original channels and channels after pruning is diplayed in the form of bar graph. It was observed that cosine and Manhttan retianed the original state at the maximum level when compared to other models.

Table 4.2.1: Accuracy and FLOPs comparison Table for cosine similarity metric

Threshold	Accuracy (in %)	Flops	Error in accuracy (in %)	Computations reduced (in %)
0.4	92.81	145943794	1.02	53.37
0.45	92.9298	163924208	0.9402	47.63
0.5	93.35	177753698	0.52	43.21
0.55	93.77	188039718	0.1	39.92
0.60	93.4	198042758	0.47	36.73
0.65	93.43	206734800	0.44	33.95
0.70	93.46	216517213	0.41	30.82

In Table 4.2.1, comparison between accuracy and flop reduction after pruning using cosine similarity criteria is performed with range of similarity threshold from 0.40 to 0.70. The maximum accuracy was obtained at threshold = 0.55. The maximum percentage of parameters were pruned at the lowest threshold of 0.4. The decrease in accuracy is 1.02 at 0.4 threshold. If compression is the main goal and compromising 1.02% of accuracy is feasible in the user's requirement, then the similairty threshold can be set at 0.40

Table 4.2.2: Accuracy and FLOPs comparison Table for euclidean distance metric

Threshold	Accuracy (in %)	Flops	Error in accuracy (in %)	Computations reduced
0.4	93.51	234257086	0.36	25.157
0.45	93.5474	243865970	0.3226	22.087
0.5	93.66	251278956	0.21	19.719
0.55	93.78	259707742	0.09	17.026
0.60	93.59	273516930	0.28	12.614
0.65	93.74	275592416	0.13	11.951
0.70	93.79	281785802	0.08	9.972

In Table 4.2.2, comparison between accuracy and flop reduction after pruning using euclidean similarity criteria is performed with range of similarity threshold from 0.40 to 0.70. The maximum Accuracy is obtained at 0.55 threshold. The maximum percentage, which was 25.157%, of parameters were pruned at the lowest threshold of 0.4. The decrease in accuracy is 0.36 at 0.4 threshold. The loss of accuracy is low

because of the lower number of parameters being pruned comapred to cosine simlarity based pruned network, which had 53.37% reduction in computation .

Table 4.2.3: Accuracy Table

Threshold	Accuracy (in %)	Flops	Error in accuracy (in %)	Computations reduced
0.4	93.58	271858423	0.29	13.144
0.45	93.62	279855938	0.25	10.589
0.5	93.75	284628611	0.12	9.06
0.55	93.7	295858518	0.17	5.476
0.60	92.68	296842113	0.19	5.162
0.65	92.65	297436394	1.22	4.9723
0.70	93.67	299349834	0.2	4.361075

In Fig 4.2.3, comparison between accuracy and flop reduction after pruning using Manhattan distance similarity criteria is performed with range of similarity threshold from 0.40 to 0.70. The maximum Accuracy is obtained at 0.5 as similarity threshold. The maximum percentage of parameters were pruned at the lowest threshold of 0.4. The decrease in accuracy is 0.29 at 0.4 threshold. The loss of accuracy is very less in this model, but the reduction in parameters is also low which means the compression of the model is less.

Table 4.2.4: Accuracy Table

Threshold	Accuracy (in %)	Flops	Error in accuracy (in %)	Computations reduced
0.4	57.5	1762620	36.37	99.436
0.45	63.2	2531536	30.67	99.19
0.5	67.8	3621618	26.07	98.842
0.55	80.35	5245318	13.52	98.324179
0.60	78.8	7534014	15.07	97.5929
0.65	78.4	11956776	15.47	96.1799
0.70	79.6	18565938	14.27	94.068375
0.75	89.7	31614248	4.1	89.899689
0.80	90.1983	52212204	3.6717	83.3187
0.85	91.5571	931275824	2.3129	70.24
0.86	92.2417	127205088	1.6283	0.0
0.87	92.1948	127205088	1.6752	59.35
0.88	93.1665	149643260	0.7035	52.19
0.89	93.0719	149643260	0.7981	52.19
0.90	93.3076	185452598	0.5624	40.749968
0.95	93.5909	262373870	0.28	16.1744

In Fig 4.2.5.,It can be seen that Reduction in accuracy is very less as the threshold increases from threshold = 0.8. The percentage of pruning is almost 40% with threshold = 0.90, maintaining the accuracy close to the original value

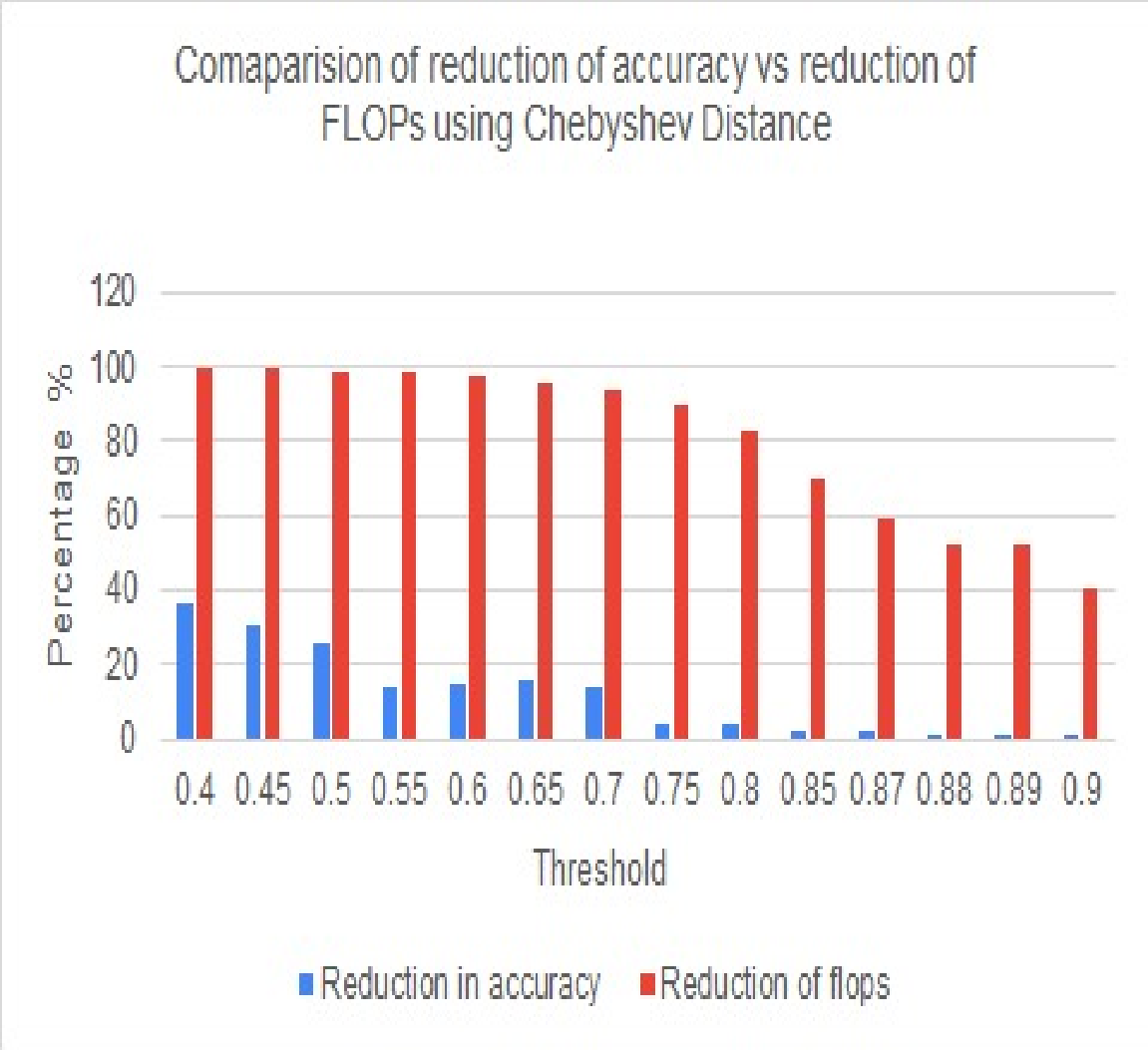


Figure 4.2.5: (Threshold vs Reduction in accuracy, Reduction of FLOPs) comparison between pruned network using chebyshev distance based similarity criteria

In Table 4.2.4, comparison between accuracy and flop reduction after pruning using chebsyshev similarity criteria is performed with range of similarity threshold from 0.40 to 0.80. The maximum Accuracy is obtained at threshold = 0.80. The maximum percentage of parameters were pruned at the lowest threshold of 0.4, but the loss of accuracy at threshold = 0.40 is 36.37% which will produce non optimal results. At the similarity threshold of 0.80 the percentage of parameters pruned is 83.3187% with very little loss of accuracy when compared to other models. Chebyshev distance based model performed decent accuracy wise at 0.80 threshold with being able to reduce 83.3187% of computation.

We can observe that Euclidean distance based model performs with highest accuracy at 0.55 threshold. But the computation reduced is more in chebyshev distance based model but there is more loss of accuracy in chebyshev distance based model when compared to other models. The chebsyshev can not be compared with other models as the FLOPs range of network pruned using chebsyshev distance is 100 times less than other models and the range of threshold in which accuracy is closer to original is different than other models. Manhattan performs decently with 5.476% of computation reduction only 0.170% loss in Accuracy.

Chapter 5

CONCLUSION AND FUTURE WORK

In this work, We implemented Similarity-aware Channel Pruning technique using different similarity criteria for compressing and accelerating CNNs. We use cosine similarity, Euclidean distance similarity, Cheybshev distance similarity and Manhattan similarity between feature maps to perform clustering and pruning for channels.

Additionally, Experiments with different threshold is conducted and graphs are plotted for the same to have a better understanding about the models. The compressed model is then retrained to reduce the accuracy loss from pruning. Euclidean distance based similarity criteria at threshold of 0.55 gave the best accuracy. The maximum number of channels were pruned when Cheybshev was used, but the accuracy produced by this model is very poor. We also analyze the impact of various similarities on the clustering in the graph in Fig. 4.2.3. Chebyshev prnued most of number of channels compressing the network the most, but the accuracy obtained by the Chebyshev model was very less to be considered ofr practical use. Cosine, Euclidean and Manhattan distance criteria have been successful in compression and acceleration of the model with minimum loss of accuracy. Tests on CIFAR-10 was carried out to see how successful these approaches are. The euclidean distance based metric performed better on VGG16 but the cosine similarity based metric performed better speed wise because of lesser FLOPs

In future, We'll combine the channel pruning strategy with other compression schemes like quantization. We'll also look into using existing techniques to speed up other real-world vision tasks, such as natural language processing. There is chance of application of this project in IoT device to integrate ML in IoT.

REFERENCES

[1] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," arXiv preprint arXiv:1608.08710, 2016.

[2] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," arXiv preprint arXiv:1808.06866, 2018.

[3] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," arXiv preprint arXiv:1611.06440, 2016.

[4] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2790–2799.

[5] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1389–1397.

[6] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in Advances in Neural Information Processing Systems, 2018, pp. 883–894.

[7] Alvarez, J.M., Salzmann, M., "Learning the number of neurons in deep networks", in: Advances in Neural Information Processing Systems, pp. 2270–2278 (2016)

[8] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers", Found. Trends® Mach. Learn. 3(1), 1–122 (2011)

[9] Wang, Zi, Li, Chengcheng, Wang, Xiangyang, and Wang, Dali. Towards Efficient Convolutional Neural Networks Through Low-Error Filter Saliency Estimation. United States: N. p., 2019. Web

[10] K. Persand, A. Anderson and D. Gregg, "Taxonomy of Saliency Metrics for Channel Pruning," in IEEE Access, vol. 9, pp. 120110-120126, 2021,

[11] Cheng, Yu Wang, Duo Zhou, Pan Zhang, Tao. (2017). "A Survey of Model Compression and Acceleration for Deep Neural Networks".

[12] Q. Zhang, Y. Shi, L. Zhang, Y. Wang and Y. Tian, "Learning Compact Networks via Similarity-Aware Channel Pruning," 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 2020, pp. 145-148

[13] OpenGenus IQ `https://iq.opengenus.org/vgg16/`

Appendix A

BIO-DATA

Name: Bhagyashri Bhamare
Address: National Institute of Technology Karnataka, Surathkal
Email: bhagyashrinilesh.181it111@nitk.edu.in
Contact No.: +91-7249504830

Name: Gagandeep KN
Address: National Institute of Technology Karnataka, Surathkal
Email: gagandeepkn.181it215@nitk.edu.in
Contact No.: +91-9986655357