Parameter Reduction in CNNs: The Role of Grouped Pointwise Convolution

Coure Project Report

CECS 551 - Advanced Artificial Intelligence

Computer Engineering and Computer Science Department

California State University, Long Beach

Fall-2023

By,

Aravind Anand (030821269)

Tapan Shah (030849791)

Instructor:

Dr. Amir Ghasemkhani

Date: December 13th, 2023

Table of Contents

1. Introduction

In the realm of machine learning, this project explores the intricate domain revolving around Convolutional Neural Networks (CNNs) and explores the realm of parameter reduction in CNN. The objective is to understand how CNN works, their architecture, and the important part they play in the interpretation of visual data. The core of the study is to implement the technique proposed for the parameter reduction and how that affects the network's efficiency. Though this project is a scope of the academic study its importance goes beyond as it can help in the development of practically useful design techniques for neural networks meant to work with real-world implementations characterized by limited computational resources. The findings explore possible modifications to the existing neural network architecture and evaluate how these adjustments perform under various controlled conditions.

2. Background

Prior to exploring the technical aspects of the project, it will be worthwhile to first understand some basic and essential concepts behind it. The background knowledge will, therefore, help in providing a clearer understanding of the project's objectives and methodologies.

2.1 Convolutional Neural Network

Deep learning for image analysis involves training machines to process visual data as humans do, and this is where Convolutional Neural Networks (CNNs) come into play. They work based on convolutions, pooling, and fully connected layers. The filters used by the convolutional layers to capture selected features, e.g. edges or textures, and pooling layers that reduce dimensional data volumes for increased efficiency and minimized overfitting. Finally, the features go through fully connected layers to classify images. In CNNs, convolutional layers are associated with weight sharing, which leads to low learning parameters making the network efficient. Their applications twerk in different fields such as video analysis and medical imaging, which makes the CNNs continue evolving, bringing machine vision closer to human vision. More information about CNN can be found in this forum.

Now let's explore the architectural design and datasets that are central to the project, examining their technical details and applications.

2.2 MobileNet CNN

MobileNet, a type of CNN, is optimized for mobile and embedded devices. It uses depth-wise separable convolutions, which significantly reduce the model's size and complexity compared to traditional CNNs. This design enables efficient performance on devices with limited computational power, without substantially sacrificing accuracy. MobileNet is ideal for various applications like image classification and object detection in environments where computing resources are constrained. Its lightweight architecture is a breakthrough in enabling advanced vision capabilities on everyday mobile devices. More elaborate information about MobileNet CNN architecture and layers can be learned in this forum.

2.3 CIFAR-10 Dataset

The CIFAR-10 dataset has been a very popular dataset in machine learning and computer vision. The dataset covers 60,000 samples and 10 classes of images with each class having an equal distribution hence resulting in a subset of about 6000 images. The dataset is divided into 50,000 training images and 10,000 test images. CIFAR-10 is simple and used as a benchmark of machine learning and image processing algorithms. For this project, the dataset was cloned from this repository of the research team.

3. Problem Statement

The depth and number of parameters contribute to the effectiveness of CNNs. The parameters in the CNNs refer to those elements of the model that can be learned with training. These are biases and weights of different layers. The number of parameters in a CNN is one of the most important criteria that affect its quality and effectiveness. In general, higher parameter values help the network in increasing its ability to understand intricate features of data. Nevertheless, a high parameter count also poses challenges. This also enhances its computational plenitude and memory needs that slow the network and make it difficult to train. Additionally, networks with so many parameters overfit and their good performance on the training data cannot be reproduced in new data never seen before.

Therefore, it is of utmost importance to manage the parameter count in CNNs. It tries to balance the network’s complexity and depth with the computational feasibility as well as generalization ability. Hence, the aim was to find resources and methodologies that could reduce overall parameters in convolutional neural

networks. In that reduction, however, we had to ensure that the performance and efficiency of the network were not compromised. The goal of the study is to explore different architectural modifications and novel approaches to make neural networks lighter without losing their computational abilities.

4. Motivation

The project is motivated by the keen interest in advancing the features of a neural network i.e. reducing the parameters count without controlling the efficiency of the architecture. The technique for optimizing neural networks in "Grouped Pointwise Convolutions Reduce Parameters in Convolutional Neural Networks", by Joao Paulo Schwarz Schuler and colleagues, introduces the concept of grouped pointwise convolution, a method that significantly reduces the parameter count in CNNs [1].

Grouped convolution in DCNN divides through the input/output channel and filter. Each set of filters can be considered a separate (parallel) route for informational flow. Rather than processing all the input sources, in a grouped convolution, every filter works with its sources of different inputs. This grouping reduces the weights present in each filter leading to the reduction of the number of floating-point computations. Notably, 1x1 filters with one trainable parameter per input channel compose pointwise convolution. Unlike (spatial) 3x3 convolutional filters, these filters do not consider surrounding positions. The technique proposed an efficient method to optimize any CNN by grouping pointwise convolutions found in its original architecture design. Also, it proposed interleaving the filters' output from separate groups at intermediate levels of successive pointwise convolutions to prevent diminishing the learning power of DCNNs. The core idea that motivated this technique is that it is unlikely that every filter depends on all output channels coming from the previous layers of the architecture. This awareness induced the support the idea that grouped convolutions can be as effective as non-grouped filters connected to all incoming channels [1].

5. Methodology

The suggested method employs a grouped pointwise convolution layer, which serves to decrease the number of parameters in the network. Here is an in-depth description of the methodology used in this approach:

- **Parameter calculation**: For a given layer 'i', the parameter count 'P_i' is computed using the formula: $P_i = C_{i-1} \cdot F_i$, where C_{i-1} is the channel count of the prior activation map, and F_i is the total number of filters in the layer.
- **Optimized Architecture**: Then, the technique introduces the grouped pointwise convolution layer followed by a channel interleaving layer. This architecture is designed to reduce the parameters while maintaining the efficiency of the architecture. The channel interleaving and grouping the output as explained in figure 5.1, aids in retaining the efficiency of the network. Diagram of the proposed pointwise convolution optimization. (Left) a classic monolithic layer M with F_i pointwise filters. (Right) our proposed replacement for M. It comprises two grouped pointwise convolutional layers (K and L) with N_i groups, where each group consists of F_i/N_i filters. H, W, and C represent the height, width, and number of channels. The size of the activation maps (represented by arrows) equals H ×W ×C. In some cases, the activation map size is divided by the number of groups N_i. The i subindex stands for the layer depth. In the case of pointwise convolutions, $H_i = H_i - 1$, $W_i = W_i - 1$, and $C_i = F_i$ [1].

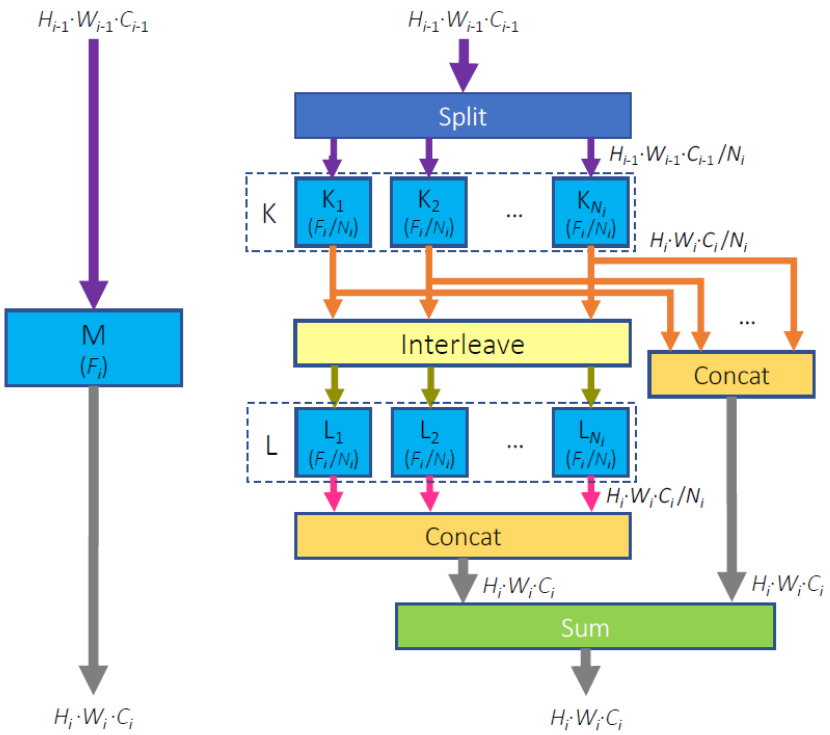

Figure: 5.1 | Grouped Pointwise Convolution Optimization [1]

- **Group Calculation:** The number of groups N_i in each layer is calculated using a specific algorithm. Each group receives a subset of C_{i-1}/N_i, input channels, with F_i/N_i filters per group.

- **Parameter Reduction Formula:** The parameters per group are determined by $(F_i/N_i) \cdot (C_{i-1}/N_i)$. The total parameters for a grouped convolutional layer are then calculated as $(F_i/N_i) \cdot (C_{i-1}/N_i) \cdot N_i$.
- **Layer Interleaving and Summation:** An interleaving layer is added between grouped convolutions to mix channels, and the outputs of both grouped convolutions are combined through summation.

These proposed methodological changes can be implemented to any CNN and the parameter count will be significantly reduced. The implementation of this methodology using Python on different variants of MobileNet version 1 architecture with CIFAR-10 dataset can be observed in python notebooks shared with the project files and the results are discussed in the further parts.

6. Experiment & Results

The proposed methodology was implemented to the MobileNet CNN architecture and processed with CIFAR-10 dataset. The details of the experimental setup and the results are discussed below.

6.1 Experimental Setup

The experiment was performed with various hardware configurations and with the aid of NVIDIA graphics card (RTX 3060 and RTX 3090). For the project, we decided to work with the Anaconda environment (with Cuda and CuDNN packages) because of its package management system and the ease it provides in setting up a development environment. This decision was particularly important as it allowed us to ensure compatibility and achieve performance when working with versions of TensorFlow and Keras. Leveraging GPU acceleration played a vital role in the project and using the precise TensorFlow and Keras versions enabled us to fully utilize the capabilities of GPU processing. As a result, we experienced improvements in the efficiency of the deep-learning computations. The details of the software, the respective versions, and details of the forums aided in the setup of the environment are provided as a read me file in the project folder. Now, let us discuss the architecture and efficiency of the network.

6.2 Architecture

The experiment was carried out with the conventional MobileNet CNN architecture and the Grouped Pointwise Convolution (GPC) MobileNet architecture to compare the parameters count and efficiency of the network. The modifications

were performed to the conventional MobileNet architecture files, such as models, layers, datasets with augmentation, and other dependencies and all the required program for the experiment is made available as a git repository to import and execute in any system. The details of the repository and project files are furnished in the read me file in the project folder.

6.3 Experimental Results and Observations

The experiment was performed in 5 different variants and the respective results of the experiment for 50 epochs can be found in Table 6.1.

Architecture	*Parameters Count*	*Parameter Reduction Percentage*	*Parameter Size*	*Computation (FLOPS)*	*Computation Reduction Percentage*	*Test Accuracy*
MobileNet	**3.239M**	NA	12.36 MB	**567.751M**	NA	**92.35%**
gpcMobileNet 16 Channels	0.274M	91.54%	1.05 MB	32.122M	94.34%	89.67%
gpcMobileNet 32 Channels	0.432M	86.66%	1.65 MB	57.813M	89.81%	90.82%
gpcMobileNet 64 Channels	0.748M	76.90%	2.85 MB	83.503M	85.29%	91.96%
gpcMobileNet 128 Channels	**1.353M**	58.22%	5.16 MB	**160.172M**	71.78%	**92.69%**

Table 6.1 | Experimental results for each scenario of MobileNet CNN and CIFAR-10 dataset

The table compares various architectures of MobileNet, a type of neural network model, based on different parameters. Here's a breakdown of each column:

- **Architecture:** Lists the different versions of MobileNet. The original version is listed as "MobileNet", followed by variants named "gpcMobileNet" with different channel sizes (16, 32, 64, and 128 channels).
- **Parameters Count:** Shows the number of parameters in each model, measured in millions (M). Parameters in a neural network are the aspects of the model that are learned from the training data.
- **Parameter Reduction Percentage:** Indicates the percentage reduction in the number of parameters compared to the original MobileNet model. This is relevant for understanding how much more efficient the newer models are in terms of parameter count.

- **Parameter Size:** Represents the size of the model's parameters in megabytes (MB). This gives an idea of the memory requirement for storing the model.

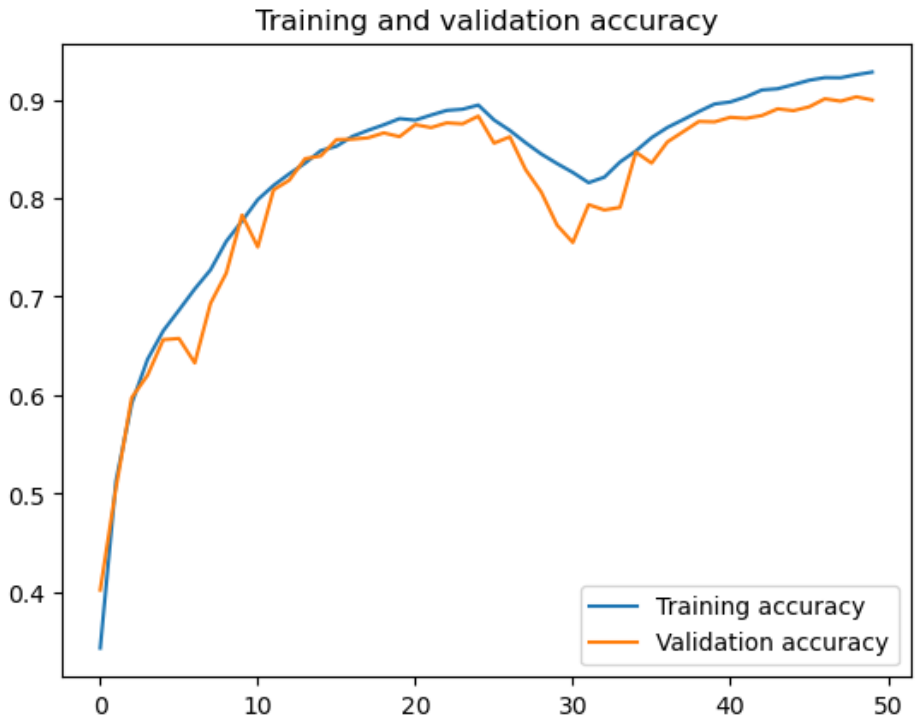

Figure 6.2 | Trend of Training and Validation accuracy over epochs

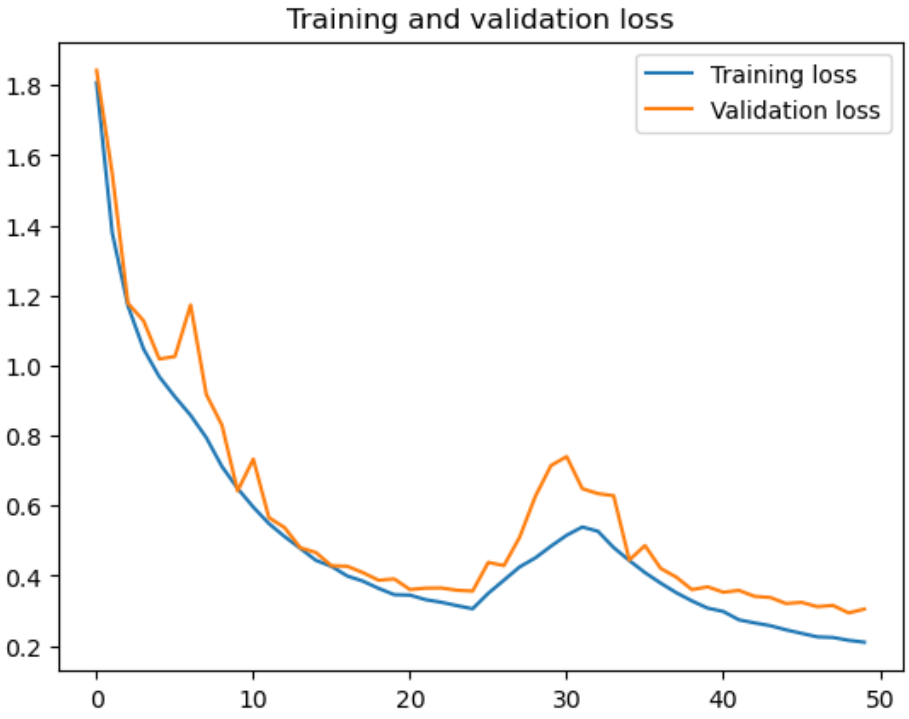

Figure 6.3 | Trend of Training and Validation loss over epochs

- **Computation:** Measured in millions (M), this column shows the computational cost of each model. It's an indicator of how much computational power is required to run the model. As part of the experiment, we could implement a function that counts the number of FLOPS (Floating Point Operations) performed during the model creation process.
- **Computation Reduction Percentage:** Like parameter reduction, this column shows the percentage reduction in computation compared to the original MobileNet. A higher percentage indicates a more efficient model in terms of computation.
- **Test Accuracy:** Represents the accuracy of each model on a test dataset, expressed as a percentage. This is a crucial metric for evaluating the performance of the model.

The architecture resulted in reducing the parameters by as much as 91.54%, while retaining up to 97.07% of the original accuracy over 50 epochs of computation. Among the various versions, the gpcMobileNetV1 with 128 channels stood out, achieving a 58.22% reduction in the number of parameters and a 71.78% decrease in the computational FLOPS count. Furthermore, the accuracy is 92.69% which is slightly higher than the standard MobileNet architecture. Even though, the GPU utilization necessitated older versions of the TensorFlow, the updated version was able to provide the parameter memory size which has significant reduction, the lower the channel count, the lower the parameter count, computations, and memory size. But, when accuracy of the network is concerned the more channel count, the better the accuracy of the CNN. The graphs illustrate the training & validation accuracy (Figure 6.2) and training & validation loss (Figure 6.3) for the optimal setup, allowing for the observation of the network's projected trends over the epochs. The accuracy and loss plots of other scenarios can be found in the python notebooks of the respective scenario.

7. Related Works

The proposed methodology in the reference paper can be applied to any Deep Convolutional Neural Networks (DCNNs) and in this project it is implemented for MobileNet version 1 for the CIFAR-10 dataset. The future goal is to implement this methodology in different DCNNs and evaluate the efficiency of the respective network. The results of the experiments conducted for other DCNNs by the research teams are discussed below. These results include the experimentation common state-of-the-art DCNNs like Efficient-Net, DenseNet, Inception V3,

MobileNet (version 1), and MobileNet V3 Large ran on different variants (channel count and input size) with different datasets like CIFAR-10, CIFAR-100, Cropped PlantDoc and Oxford-IIIT Pet testing.

The results show that DCNNs with the proposed technique when trained from scratch, obtained similar test accuracies to the original EfficientNet and MobileNet V3 Large architectures while saving up to 90% of the parameters and 63% of the floating-point computations. For instance, the modified EfficientNet-B0 32 channel variant achieved a slightly higher classification accuracy than the EfficientNet-B0 baseline on the CIFAR-10 dataset, with 26% of the original parameters and 45% of the original computations.

In the case of the Cropped-PlantDoc dataset, all the modified variants achieved higher test accuracies than their baselines after 75 epochs. The modified MobileNet V3 Large 16ch variant had only 10% of the original trainable parameters, 37% of the original computations, and achieved higher accuracy by a margin of 15%. The modified EfficientNet 16 channel variant had 16% and 33% of the original trainable parameters and computations, respectively, and achieved a higher accuracy than its baseline by a margin of 0.5%

The research team also presents Class Activation Maps (CAMs) for the Oxford-IIIT Pet dataset, which show that the modified EffNet-B0 focuses more on the relevant regions of the images compared to the EfficientNet-B0 baseline.

8. Conclusion

In this project, the Grouped Pointwise Convolution was implemented in the MobileNet model to show that the use of the proposed methodology will reduce the parameters count and computation complexity without compromising on the accuracy levels of the network. Results from the experiment indicated balance in the dual approach depending on different channel counts and the leaning of the accuracy of the network, i.e. with low channels reducing parameter count and memory size and more channels improving accuracy. The experimental results of the study such as training/validation accuracy and loss are vital in providing an understanding of the network's progress over epochs. Further works might apply these methodologies to other deep convolutional neural networks architectures and evaluate the performance of those modified networks, helping to advance the research on efficient neural network design.

9. References / Useful Links

[1] Joao Paulo Schwarz Schuler et al. “Grouped pointwise convolutions reduce parameters in convolutional neural networks”. In: Mendel. Vol. 28. 1. 2022

[2] Convolutional Neural Network – by IBM

[3] An overview of MobielNet Architecture – Medium Post by Srudeep PA

[4] CIFAR-10 Dataset by Alex Krizhevsky

[5] Install TensorFlow, CUDA and CuDNN with pip - Tensorflow Forum