CNN model compression using Genetic Algorithm

A
Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY IN ARTIFICIAL INTELLIGENCE
By

RAKESH KUMAR LAL
Under the Supervision of
Dr. SHASHI SHEKHAR JHA
Dr. SUDEEPTA MISHRA

Department of Computer Science & Engineering
Indian Institute of Technology Ropar
May 9, 2024

Declaration

This is to certify that the thesis entitled "**CNN model compression using Genetic Algorithm**", submitted by me to the *Indian Institute of Technology Ropar*, for the award of the degree of Master of Technology in Artificial Intelligence, is a bonafide work carried out by me under the supervision of Dr. Shashi Shekhar Jha, and Dr. Sudeepta Mishra. The content of this thesis, in full or in parts, have not been submitted to any other University or Institute for the award of any degree or diploma. I also wish to state that to the best of my knowledge and understanding nothing in this report amounts to plagiarism.

Sign: ______________________________

Rakesh Kumar Lal
Department of Computer Science & Engineering,
Indian Institute of Technology Ropar,
Rupnagar-140001, Punjab, India.

Date: ______________________________

Certificate

This is to certify that the thesis entitled "**CNN model compression using Genetic Algorithm**", submitted by Rakesh Kumar Lal (2022AIM1011), a master's student in the *Department of Computer Science & Engineering*, *Indian Institute of Technology Ropar*, for the award of the degree of Master of Technology in Artificial Intelligence, is a record of an original research work carried out by the candidate under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in my opinion has reached the standard worthy of the award of the degree. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Sign: ______________________________

Supervisor: Dr. Shashi Shekhar Jha, and Dr. Sudeepta Mishra
Department of Computer Science & Engineering,
Indian Institute of Technology Ropar,
Rupnagar-140001, Punjab, India.

Date: ______________________________

Acknowledgements

I would like to express my deepest gratitude to all those who have supported and guided me throughout the course of this endeavor.

First and foremost, I extend my heartfelt appreciation to my supervisors, Dr. Shashi Shekhar Jha and Dr. Sudeepta Mishra for their invaluable guidance, unwavering support, and insightful feedback throughout this project. Their expertise and encouragement have been instrumental in shaping the outcome of this work.

I am immensely thankful to my colleagues and peers for their camaraderie, stimulating discussions, and collaborative spirit, which have enriched my understanding and inspired new ideas.

I am indebted to my friends and family for their love, encouragement, and understanding during the highs and lows of this journey. Their unwavering belief in me has been a constant source of strength.

Finally, I would like to express my gratitude to Department of Computer Science & Engineering at Indian Institute of Technology Ropar for their resources, which made this work possible.

To all those mentioned and those unnamed but not forgotten, thank you for being part of this journey and for your invaluable contributions.

Sincerely
Rakesh Kumar Lal

Abstract

Convolutional Neural Networks (CNN) are important part of many Computer Vision algorithms, but due to their large size and slow inference speed, they cannot usually be used with resource-limited devices. This work presents an attempt to solve these problems by pruning redundant or less contributing filters from the convolutional layers of the given CNN model by using Genetic Algorithm (GA). Our work mainly focuses on sequential CNNs. The two important metrics used to evaluate a pruned CNN model are (i) accuracy drop, and (ii) number of parameters dropped, both relative to the original model. Our algorithm tries to remove much of the less contributing filters while maintaining the accuracy drop of the pruned model under an acceptable threshold. Our results have shown that significant amount of such filters can be removed from the original model without retraining the pruned model.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

THIS chapters introduces the relevant topics and motivation behind our work.

1.1 Motivation

After AlexNet won the 2012 Imagenet challenge, deep convolutional neural networks quickly gained popularity. Now it is the workhorse of many Computer Vision algorithms. Many researches have also led to different types of architectures and optimizations methods. There are several architectures of convolutional neural networks that have been carefully designed to solve various problems. These CNNs have deep convolutional layers each with certain number of filters. Given a trained model based on some architecture it might be the case that not all the filters in the model are contributing significantly during inference. There might be many redundant filters that are not contributing much or are also causing over parameterization of the model. Since convolution operations are expensive and slow, it is desirable to remove such filters which would also reduce the number of parameters.

1.2 About pruning parameters

In [1], Le Cun *et al.* showed that Optimal Brain Damage (OBD) can be used to selectively eliminate parameters from a neural network model without significantly affecting its performance. In CNN input feature extraction is performed by the convolutional filters and it seems reasonable that the idea of OBD can be used to eliminate less essential filters.

Removing filters from a CNN is a form of structured pruning where rather than individual parameters, entire set of parameters in the form of filters or layers are removed. A model pruned this way does not require special data structure or algorithms to operate and are generally consistent with existing neural network frameworks like tensorflow and torch.

By removing filters from a CNN model we can impact several performance metrics of the model like (i) parameters count, (ii) Floating Point Operations per sec (FLOPS) count, (iii) accuracy, and (iv) inference time.

1.3 Problem statement

The main idea we use for pruning is to assign a bit to each filter in the given model such that bit 0 would mean that the filter is less contributing and can be removed, whereas bit 1 would mean that the filter is important and must be preserved. Now the problem is to find a sequence of bits such that the pruned models have most of its less desirable filters removed while also maintaining the accuracy drop within a threshold value.

The search space for this problem is exponential in number of filters and an algorithm that can give good solution in huge search space is needed.

1.4 Genetic Algorithm

Genetic Algorithm is an Evolutionary algorithm that is inspired by the nature. It is an iterative and progressive optimization algorithm that keeps improving the solution that it has found on each iteration. It is based on the principle of natural selection i.e the individuals that have higher fitness score have better chances of participating in gene sharing. Genetic Algorithm can finds good sequence of genes such that the individual that the DNA represents have desired traits defined by some fitness function. This algorithm is generally used to solve problem that have very large search space and is suitable for finding solution to our problem. We will examine Genetic Algorithm in detail in Chapter 3.

Chapter 2

Related Works

THIS chapter outlines the works and methods related to pruning or reducing the parameters of Neural Networks.

2.1 Parameter reduction approaches

2.1.1 Architecture Search

In this approach the network is built from scratch by using trial and error search to find good architectures. Methods like [2] Reinforcement Learning, [3] [4] Evolutionary algorithms, [5] Hill Climbing, etc. are used to find good designs for the model. Since the design space of CNNs can be very large, these algorithms are usually very slow and consumes lots of GPU time to train. Our work differs from this approach as we are already provided with a trained model and our objective is to eliminate filters from it.

2.1.2 Knowledge Distillation

[6] [7] This approach involves making a smaller neural network called student network to learn the response-behaviour of a larger network called teacher network. It is a process of transferring knowledge from a larger network to a smaller one.

2.1.3 Network Pruning

[8] Network pruning deals with removing parameters (which may entail removing individual parameters, or parameters in groups such as by neurons) from an existing network. The goal of this process is to maintain accuracy of the network while increasing its efficiency. This can be done to reduce the computational resources required to run the neural network. To determine the importance of a parameter the model needs to be evaluated before and after removing that parameter and comparing the differences. Our approach comes under this category.

Chapter 3

Method

THIS chapter demonstrates our approach on pruning the less significant filters from the given model using Genetic Algorithm.

3.1 Metrics for model evaluation

We have used two metrics, accuracy drop and number of parameters dropped to evaluate pruned models. Evidently these two metrics are positively correlated. Reducing the number of parameters causes increase in accuracy drop. We want our algorithm to maximize the drop in parameter count while limiting the accuracy drop to an acceptable threshold value as per the requirement. Note that we are not trying to optimize the accuracy drop because then the algorithm will try to balance both these metrics which might not be what we want. By limiting the accuracy drop to a threshold value we are effectively informing the algorithm about how to determine which filter is good for pruning and which one is not. Those filters whose removal causes the accuracy drop to exceed the threshold are good filters (bad pruning) whereas those filters after whose removal the accuracy drop remains in the acceptable region are bad filters (good pruning).

3.2 Genetic Algorithm working

Genetic algorithm is an iterative algorithm as shown in 4.6. It begins with randomly initialised population. Every individual in the population is evaluated using a fitness function. Individuals are then selected for crossover based on some selection criteria. After this crossover operation is performed between the selected individuals to obtain a list of offspring. The offspring are then mutated with some probability i.e. some of their bits are flipped. The mutated offspring are then added to the population. Population can be thresholded based on fitness score of the individuals if storage is limited. The loop continues until some stopping criteria is met, which most often is either when the individuals converges to solution or until max generation has reached.

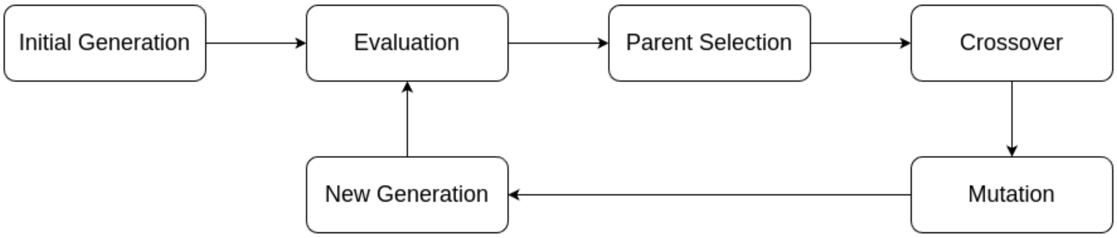

Figure 3.1: Genetic Algorithm cycle

All applications of Genetic Algorithm uses same flow of operations but their difference may lie in their implementations.

3.3 Our implementation

3.3.1 Initial population

Instead of random initial population all individuals in our implementation are comparable to the original model. We maintain a pointer to the individual with highest score in current generation, called BEST. We construct the fitness function in such a way that BEST would always point to an individual within acceptable accuracy region. So after every generation the BEST will either point to the best individual of the previous generation or to an even better individual found in the current generation. So effectively BEST always points to the best individual in the acceptable accuracy drop region throughout all the generations so far. After termination of the algorithm the individual that BEST points to is returned as a solution.

3.3.2 Parent selection

We are free to use any approach for parent selection. In our experiment we used some number K of top individuals that have highest fitness score in the population to participate in crossover. This approach converges fast to a good solution but loses population diversity rapidly. There are other methods like [9] FUSS that could maintain population diversity over a long run of the algorithm.

3.3.3 Crossover

Genetic algorithm is a search algorithm. More the number of individuals in the population, more its searching capability will be. When individuals finds beneficial updates to the bit sequence they share that information with each other during the crossover phase. This is generally why only the best individuals in the population are allowed to participate in this process.

There are several methods for crossover like single point, double point and uniform crossover. In our approach we have used single point crossover.

In single point crossover a random crossover point is selected and the bit sequence on either side are recombined to obtain two offspring. So in total four offspring are returned in this process two of them are re-combinations of the parent whereas the remaining two offspring are identical copy of the two parents.

3.3.4 Mutation

Mutation helps in the searching process. Genetic algorithm tends to converge to a local optimal solution. Mutation allows the algorithm to get unstuck and converge towards a global solution and so is an important step in the algorithm.

3.3.5 Fitness function

As mentioned earlier, we need to construct the fitness function in such a way that the best individual always remains within the provided threshold value of accuracy drop while pruning removes filters. This can be considered as a constrained optimization where we put constraints on one metric while optimizing the other.

Algorithm 1 Fitness function

```
Input: individual, threshold
Output: individual
  accuracy_drop ← individual.accuracy_drop(%)
  parameters_drop ← individual.parameters_drop(%)
  individual.score ← 0
  if accuracy_drop < threshold then
      individual.score ← parameters_drop          ▷ accuracy is in an acceptable region
                                                  ▷ more parameter drop gives more score
  else
      individual.score ← $\frac{\text{parameters\_drop}}{(\text{accuracy\_drop}+\epsilon)} - \lambda \times \text{accuracy\_drop}$
                                                  ▷ where $0 \leq \lambda \leq 1$ and $\epsilon \in \mathbb{Z}^+$ is an offset
  end if
```

The idea behind this fitness function is that if accuracy drop of the individual is below the threshold then we evaluate the individual only on the parameter drop metric because the other metric, accuracy drop is already satisfactory. If not then we evaluate the individual on both the metrics. The score then must be directly proportional to the parameters dropped and inversely proportional to accuracy drop. We note that accuracy drop can also take negative values, when algorithm removes filters that were causing over-parameterization, in which case the proportionality gets inverted. To solve this issue we add an offset ϵ to accuracy drop in denominator. The value of offset should be large enough to consider the largest negative drop in accuracy. A value of 10 for ϵ should be sufficient. We also note that if we just consider this ratio as score then for same ratio multiple pairs of numerator and denominator are possible. We would like to have the pair with least accuracy drop to have a higher score than those pairs with relatively higher drop. To achieve this we deduct a weighted amount of accuracy drop from this ratio to obtain the score.

3.3.6 Genetic Algorithm

Algorithm 2 presents the steps in our Genetic Algorithm.

Algorithm 2 Genetic Algorithm

```
Input: I=initial population count, K=number parents, P=mutation probability,
  T=population threshold, M=max generations
Output: BEST
  Population ← I individuals close to original model
  BEST ← null
  Gen count ← 0
  while Gen count $\leq$ M do
      Evaluate population.
      Parents ← Select K individuals with highest fitness score for crossover.
      offspring list ← Crossover(Parents)
      for offspring $\in$ offspring list do
          mutate offspring with probability P
      end for
      Evaluate offspring list.
      population ← population + offspring list
      threshold population by keeping only T top individuals.
      BEST ← individual with highest fitness score in population.
      Gen count ← Gen count + 1
  end while
```

3.3.7 Our settings for experiment

Our experimental setup.

I:	50
K:	10
P:	0.0002
T:	300
threshold:	2

Table 3.1: Settings of different parameters used in our experiment

3.3.8 Base models and model evaluation

We have used VGG16 trained on CIFAR10 dataset and VGG19 trained on SVHN dataset to prune using this algorithm.

For evaluation of the individuals in each generation we used entire training dataset. We noted that by doing so the testing accuracy drop is comparable to the training accuracy drop for the pruned models.

Chapter 4

Results

THIS chapter shows results of our experiment.

4.1 Pruning VGG16 trained on CIFAR10 dataset

4.1.1 Accuracy drop

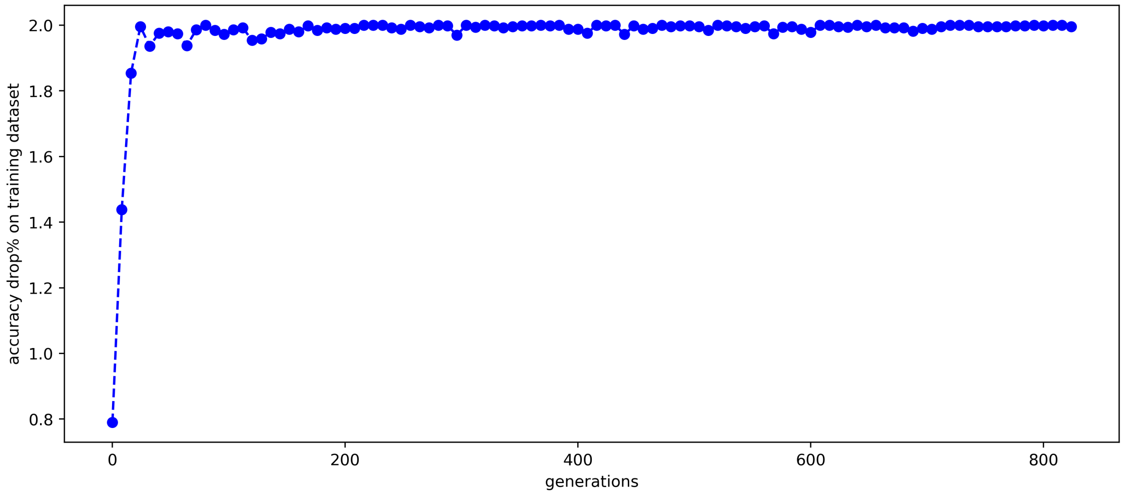

FIGURE 4.1: (VGG16, CIFAR10): Accuracy drop% of BEST individual on training dataset

As algorithm prunes filters from the individual pointed by BEST, its accuracy drop also decreases. The BEST always points to an individual with accuracy drop within the threshold value.

4.1.2 Parameters drop

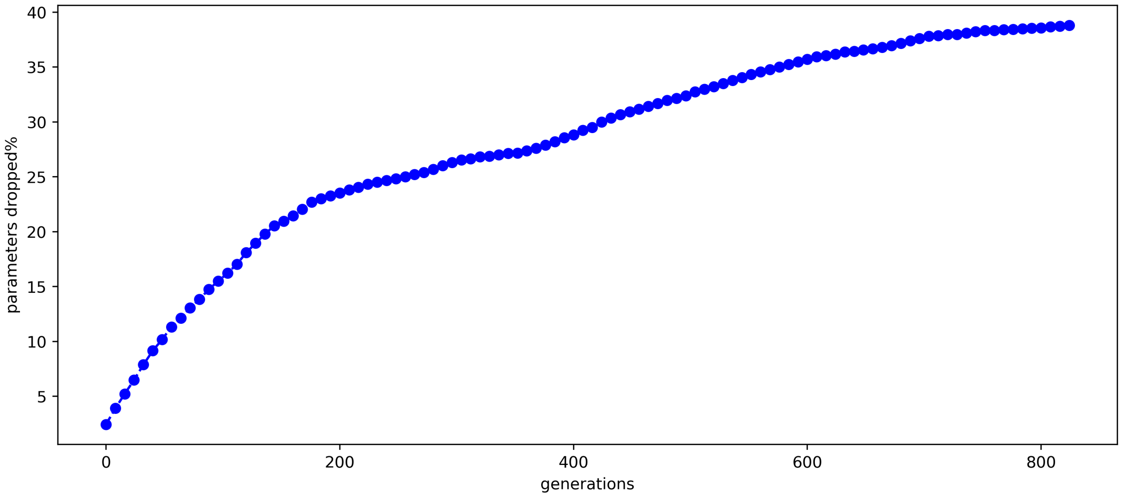

Figure 4.2: (VGG16, CIFAR10): Parameters dropped% of BEST individual over generations

Parameters drop increases with increase in the number of pruned filters.

4.1.3 FLOPS drop

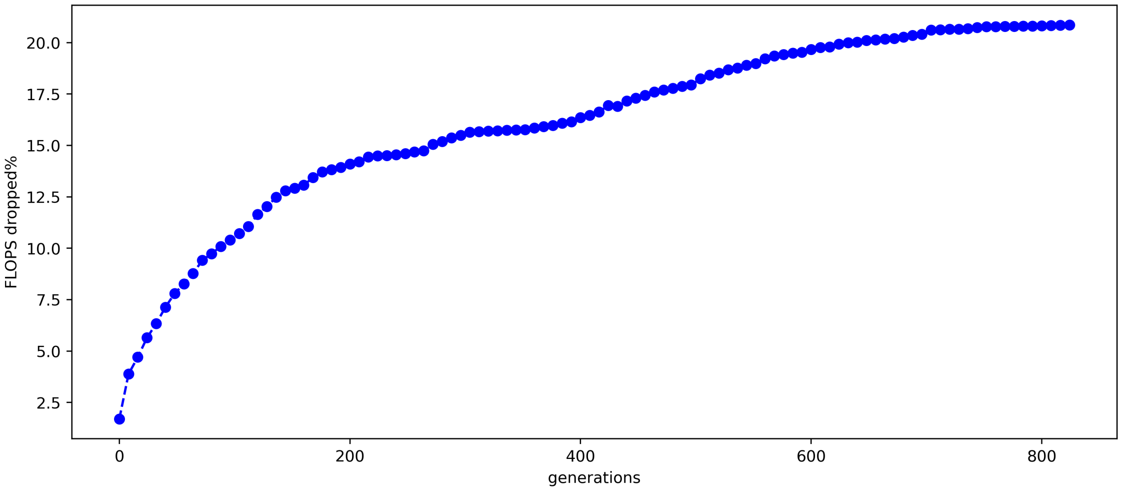

Figure 4.3: (VGG16, CIFAR10): FLOPS dropped% of BEST individual over generations

Floating Point Operations Per Sec (FLOPS) measures the number of floating-point calculations that can be performed in one second. We can see that its plot have same trends as that of parameters drop.

4.2 Pruning VGG19 trained on SVHN dataset

4.2.1 Accuracy drop

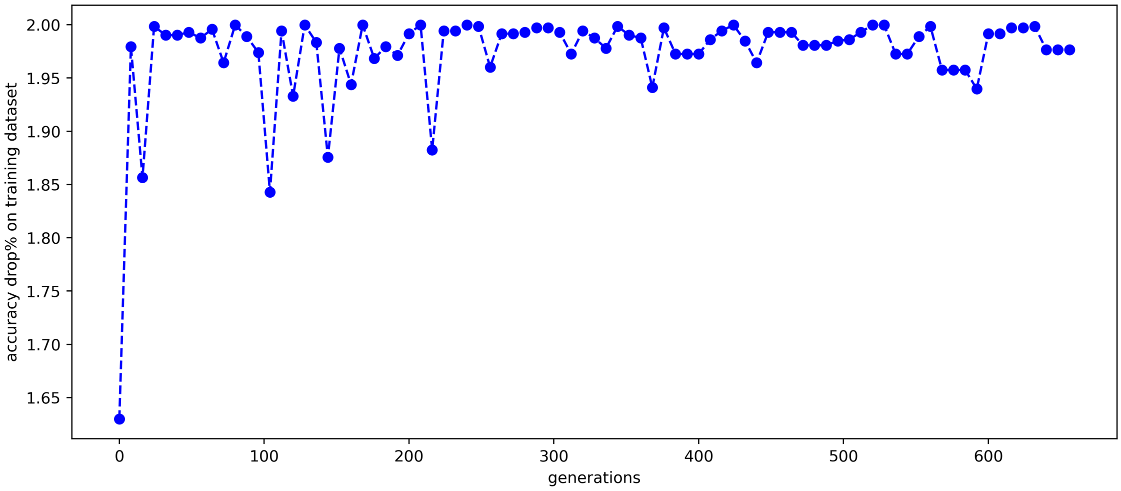

FIGURE 4.4: (VGG19, SVHN): Accuracy drop% of BEST individual on training dataset

4.2.2 Parameters drop

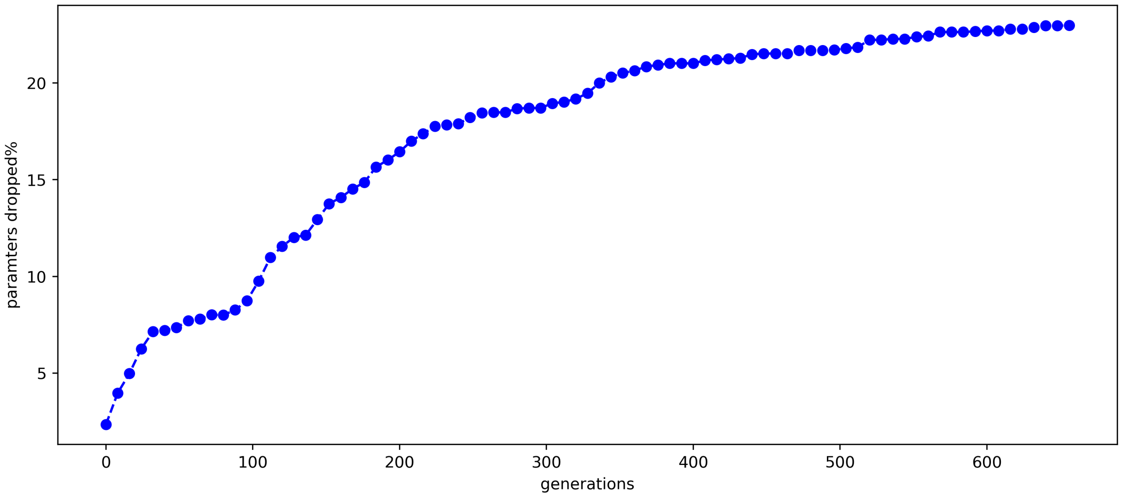

FIGURE 4.5: (VGG19, SVHN): Parameters dropped% of BEST individual over generations

4.2.3 FLOPS drop

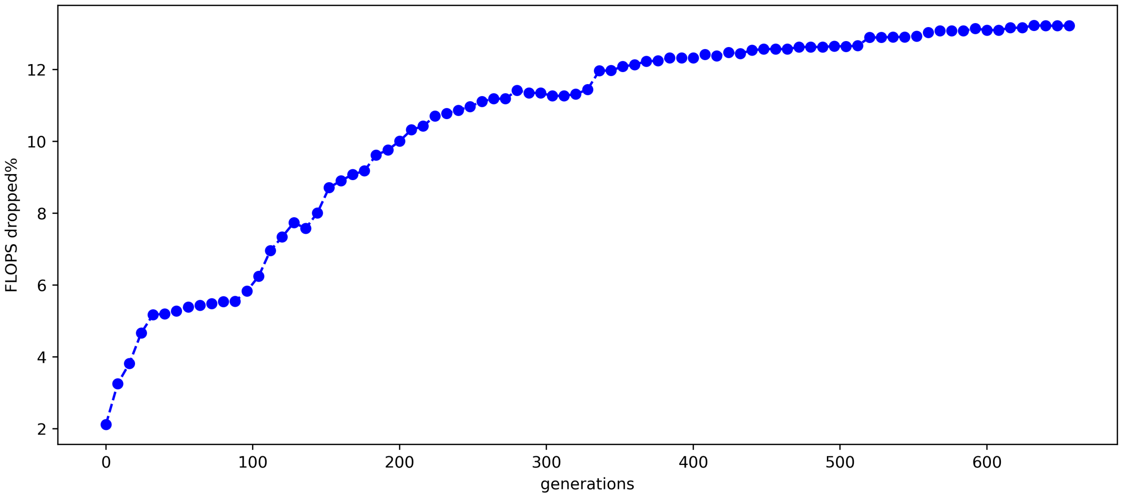

FIGURE 4.6: (VGG19, SVHN): FLOPS dropped% of BEST individual over generations

We observe that all the three plots have similar trends.

4.2.4 Comparing the best solution with base model

Now we compare the pruned model.

VGG16 on CIFAR10 dataset		
Metrics	Base model	Pruned model
Test accuracy	86.87	85.24 (1.87% drop)
Number of parameters	15245130	9326336 (38.82% drop)
FLOPS count	313725952	252446852 (19.523% drop)

TABLE 4.1: (VGG16, CIFAR10): Comparing metrics of pruned model and base model

VGG19 on SVHN dataset		
Metrics	Base model	Pruned model
Test accuracy	92.43	91.418 (1.09% drop)
Number of parameters	20554826	16235999 (21.011% drop)
FLOPS count	398660608	349536612 (12.32% drop)

Table 4.2: (VGG19, SVHN): Comparing metrics of pruned model and base model

Chapter 5

Conclusion and scope for further research

We showed a method to eliminate filters that are either causing over-parameterization of the trained model or are very insignificant while making predictions. If we set the threshold to 0 then the returned model have negative test accuracy drop i.e. the pruned model performs even better than the base model on test dataset.

Our work attempted to remove these filters from the base model without re-training the pruned models. We suspect that re-training the pruned model and repeating the algorithm could allow for pruning even more filters.

Bibliography

[1] Y. L. Cun, J. S. Denker, and S. A. Sol1a, “Optimal brain damage.” in *Advances in Neural Information Processing Systems 2 (NIPS 1989).* AT&T Bell Laboratories, Holmdel, N. J. 07733, 1989, p. 604.

[2] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” 2004.

[3] M. Suganuma, S. Shirakawa, and T. Nagao, “A genetic programming approach to designing convolutional neural network architectures,” 2003.

[4] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat, “Evolving deep neural networks,” 2004.

[5] E. Thomas, M. Jan Hendrik, and H. Frank, “Simple and efficient architecture search for convolutional neural networks,” 2013.

[6] G. Hinton, O. Vinyals, Dean, and Jeff, “Distilling the knowledge in a neural network,” 2015.

[7] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, “Learning efficient object detection models with knowledge distillation,” in *Advances in Neural Information Processing Systems*, 2017, pp. 742–751.

[8] “Pruning (artificial neural network),” in *Wikipedia*, April 2024.

[9] H. Marcus, “Fitness uniform selection to preserve genetic diversity,” 2002.