MINI PROJECT 4 - COMP 551

Saad Shahbaz : 260845253,
Ibrahim Naveed : 260857811,
Sym Piracha : 260846061,

A project presented for the class of
COMP 551 : Machine Learning

Department of Computer Science
McGill University
Quebec,Canada
13th December, 2021

1 Abstract

In this project, we compare the SqueezeNet architecture with a number of popular architectures, as well as our own modified SqueezeNet, on the CIFAR10 and the Fashion-MNIST datasets. We compare the models based on accuracy giving importance to model size since this is integral for better deployment of AI in our everyday lives. Our results found that SqueezeNet was able to perform similarly to other much larger CNN architectures such as AlexNet with 78 times less parameters. Our customized CNN 'ISS' was able to emulate this with minor modifications showcasing a systematic approach to building CNN architectures.

2 Introduction

MiniProject 3 sparked our curiosity on how CNN architectures are designed and all three of us have several questions we wanted to explore. We thought to ourselves: how are design decisions made in the real-word? Is the process of CNN design only dictated by trial and error? Are there well-defined architecture types that are preferred over others for certain machine learning problems? Can there be a systematic approach to designing such architectures? These were the main questions we had in mind as we decided what paper to choose for this project.

Dozens of well-documented and reliable CNN architectures for Image Recognition have been proposed over the years. Some notable ones include AlexNet[1], GoogleLeNet[2] and VGG[3]. These are incredibly complex and accurate models with similar building components like convolution and pooling layers with some topological differences. They differ slightly in the types of problems they are used to solve. For example, GoogleLeNet has been shown to outperform VGG in large-scale data analysis, whereas the latter generalizes better over a large variety of datasets. One aspect we saw that was common between all these architectures was the vast number of parameters that had to be optimized during training.

From our research we concluded that the two most important things to look at when it comes to choosing which model to pick for your problem are the size (number of parameters) and the accuracy of the model in question. An ideal model would have a small size while remaining accurate. Traditionally, there has always been a tradeoff between these two. Decreasing the size of a model is followed by a decrease in accuracy. For instance, AlexNet is an incredibly accurate model, however, it has 61 million parameters[4] making it difficult to be deployed in real world applications.

2.1 SqueezeNet

The paper we found was proposed in 2017 and introduced the SqueezeNet Architecture. SqueezeNet promised AlexNet-level accuracy on the ImageNet dataset with 50x fewer parameters and 510x smaller size with compression[5]. Squeeze net achieves this decreased size by following three design strategies. Firstly, it replaces 3x3 filters with 1x1 filters which contain 9x less parameters. Secondly, it decreases the number of inputs to 3x3 filters. Thirdly, it downsamples late in the network in order to ensure that the convolution layers have large activation maps. This maximizes accuracy on a limited budget of parameters. SqueezeNet makes use of basic building blocks which it calls "Fire Modules" as shown by figure 1 in the appendix. These modules consist of a "squeeze" convolution layer, which has 1x1 filters, passed into an "expand" layer that has a mix of 1x1 and 3x3 convolution filters. The output of these layers are concatenated and then the ReLU activation function is applied before sending this output to the next Fire Module. The final CNN architecture contains a stand-alone initial convolution layer, followed by 8 Fire Modules, and ends with another final convolution layer. Squeeze Ratio is used as a hyperparameter to configure the number of filters in squeeze layers versus the number of filters in the expand layers. This allows us to change the number of parameters in the model easily and observe the subsequent changes in accuracy this produces. Max pooling with stride is performed on several occasions throughout the network and the architecture also makes use of bypass connections which takes inspiration from the ResNet[6] architecture.

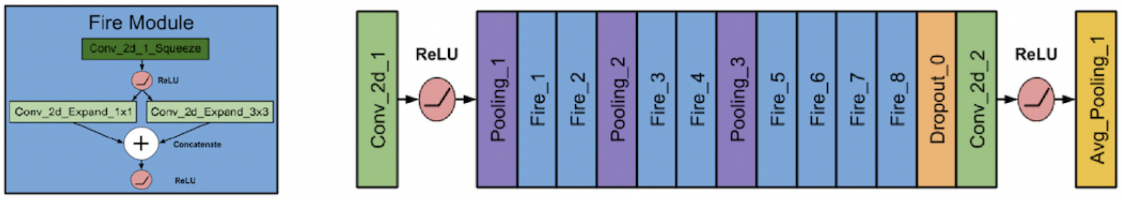


Figure 1: Fire Module and Squeezenet Architectures

3 Experiments

The central claim of SqueezeNet is that it is possible to achieve accurate results compared to other well established models while keeping model size minimal. We wanted to test if these claims hold true not only in comparison to AlexNet but also to models such as ShuffleNet[7]. In addition to this, we included a slight variation to the original SqueezeNet architecture which we call "ISS". Furthermore, we included the CNN model we had in our MiniProject-3 to get a better understanding of how our model would perform on different datasets relative to these established models.

In the paper, a comparison was made between SqueezeNet and AlexNet using the ILSVRC 2012[8] dataset. In our experiments, we were interested in observing how our 5 different models would generalize over two different datasets: CIFAR-10[9] and Fashion-MNIST[10]. The CIFAR-10 dataset consists of 60,000 32x32 color images with 10 classes such as bird, dog, cat, car. The Fashion-MNIST dataset contained 70,000 examples of 28x28 greyscale images of 10 different clothing items such as bags, sneakers, and shirts. Our reasons for choosing these two datasets were threefold. Firstly, we wanted to investigate how accurate these models would be on data they were not originally designed for. Secondly, it was important to us to evaluate multiple datasets as one model might have a certain inductive or learning bias towards a specific kind of data. Thirdly, by including one RGB dataset and one Greyscale dataset we can observe if there are significant differences in the performance of our models. This decision was made to add another dimension of evaluation metrics when comparing our models.

A lot of the experimentation revolved around creating our own variation of SqueezeNet. We tested to see if increasing the size of our models slightly would lead to a further increase in performance. As a result, we added two additional linear layers at the end of the architecture. This substantially increased training accuracy (92.5% to 96.4%) hinting towards overfitting, therefore, we decided to use dropout as a regularization technique. All combinations of dropouts between 0.1 and 0.5 were tested for these two linear layers.

To ensure a fair study of the different models, based on preliminary tests on a small subset of random values, we fixed our hyper-parameters, namely: learning rate, batch size, epoch size and momentum. These values are shown in figure 3 in the appendix. This enabled us to standardize our final results and present them in a way that allows us to make impactful observations on different models.

4 Data Processing

Both datasets were resized to 224x224 images to meet the input criteria of our pretrained models. For the Fashion-MNIST dataset grayscale values had to be repeated across the 3 channels to allow for a more fair comparison to the CIFAR-10 dataset. Images were normalized by subtracting the mean from each pixel and then dividing the result by the standard deviation to center the data around zero mean for each channel.

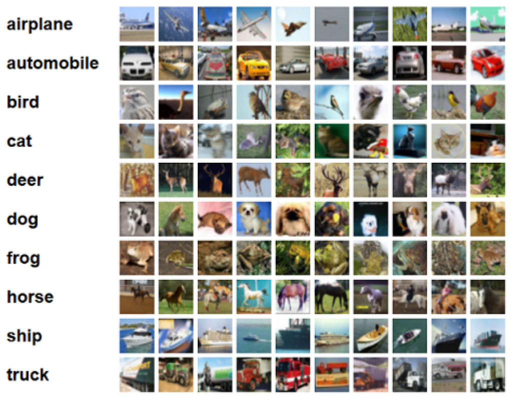


Figure 2: Sample images from the CIFAR10 dataset

5 Results

Model	CIFAR-10 Acc	MNIST Acc	Model Size (of parameters)
SqueezeNet	90.5%	92.6%	727,626
AlexNet	91.1%	93.2%	57,044,810
ShuffleNet	90.4%	93.1%	3,520,420
MiniProject3	25.3%	67.9%	4,073,854
ISS	89.8%	93.7%	797,066

Our results go hand in hand with the results of the research paper. As shown in the table, SqueezeNet has 78 times less parameters yet achieves an almost equivalent accuracy rate compared to AlexNet. High levels of accuracy across both datasets also show that SqueezeNet generalizes well. Comparing SqueezeNet to another well researched architecture such as ShuffleNet, we find that SqueezeNet again performs very similarly in terms of accuracy. For our custom architecture, ISS, we found that the dropout value of 0.1 gave us the best test accuracy as seen from figure 2 in the appendix. Our slight increase in parameter size allowed ISS to outperform all other architectures we looked at for the MNIST dataset. To our surprise, Mini 3 performed poorly on both the datasets, especially on the CIFAR 10 dataset.

6 Discussion and Conclusion

From our results we can see that it is possible for a model with comparatively few parameters to perform as well as an industry level architecture. Our experiments on the ISS are promising as they point to the idea that one can start off with a small model and incrementally add layers while monitoring the impact of the increased size on the accuracy. This allows for a customised and controlled designing paradigm depending on the size requirement of the project. It is important to realize that we fixed our hyper parameters. It is entirely possible that each architecture has a different configuration of hyperparameters which would optimize its performance. We suspect that the low performance of Mini 3 is due to the fact that it was a very bounded architecture designed solely for recognizing handwritten digits and alphabets. This may be why it yields a high generalization error.

For future work, the authors of SqueezeNet propose a hyperparameter called “Squeeze Ratio” that would allow easier modifications of model size. We would be interested in seeing how SqueezeNet compares to other models such as ShuffleNet and ISS as we tweak this ratio.

The traditional approach of focusing solely on accuracy is outdated as mentioned by Vincent Houdebine who describes how the average android application is 15MB whereas NASNet, a state of the art image classification model, has a 355MB size.[11] In conclusion, we believe that SqueezeNet is a beneficial step forward in recognizing the importance of model size. Lightweight models help facilitate the deployment of machine learning solutions in our ever-increasing automated lifestyle.

7 Statement of Contributions

All members contributed to the best of their abilities.

References

[1] Visualizing and comparing alexnet and vgg using deconvolutional layers.

[10] Fashion-mnist dataset.

[11] Making neural networks smaller for better deployment.

[2] Going deeper with convolutions.

[3] Very deep convolutional networks for large-scale image recognition.

[4] Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size.

[5] Imagenet classification with deep convolutional neural networks.

[6] Deep residual learning for image recognition.

[7] Shufflenet: An extremely efficient convolutional neural network for mobile devices.

[8] Ilsvrc 2012 dataset.

[9] Cifar-10 dataset.

8 Appendix

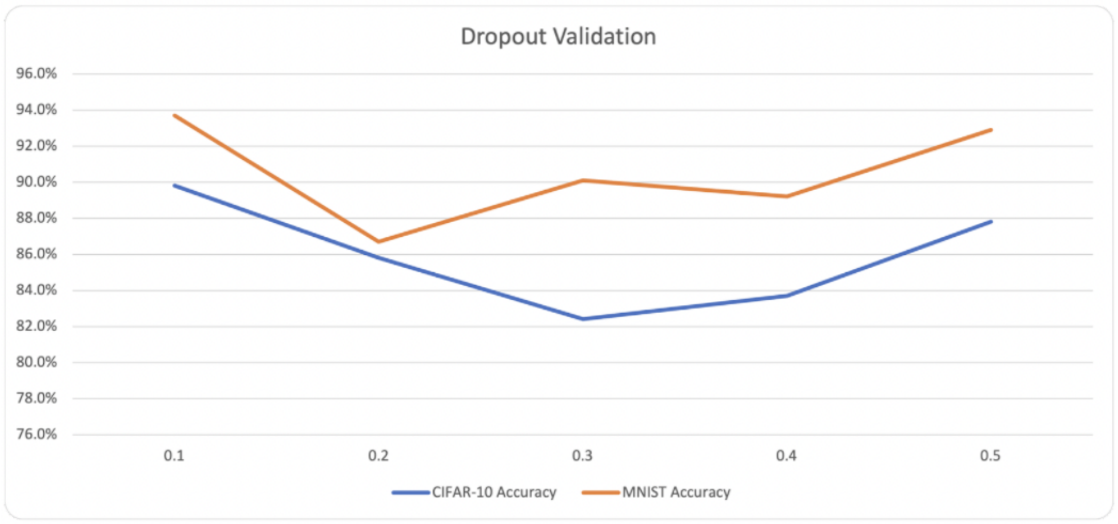
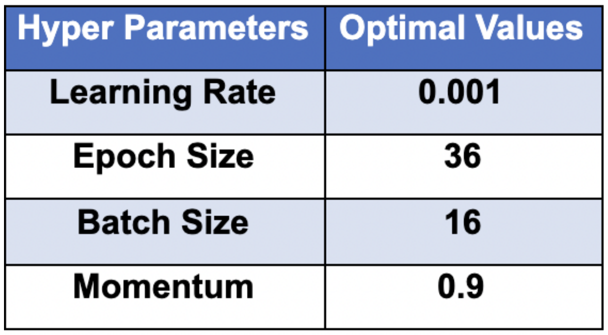


Figure 3: Accuracy



Hyper Parameters	Optimal Values
Learning Rate	0.001
Epoch Size	36
Batch Size	16
Momentum	0.9

Figure 4: Hyperparameters