Image Classification Using Convolutional Neural Networks

Priscilla M. Babiak, Hugo A. Lara, *ECE 523, University of Arizona*

Abstract An image classification model requires a different approach to training neural networks because retaining spatial information must be taken into consideration. The CIFAR dataset was used to employ a deep neural network used for image classification. For this, Convolutional Neural Networks were coupled with other strategies such as adding dropout, batch normalization, and ResNet in order to explore the best performing model. While the convolutions applied to each layer of the model seek to learn different attributes of the image, such as edges, too many layers can cause it to degrade due to the vanishing gradient problem which plagues deep neural networks –therefore, ResNet was applied. Issues with overfitting were explored by reducing the percentage of dropout used at each layer and exploding gradients were realized when batch normalization was removed from the network. It was found that there is no deterministic way to fit such models, and multiple configurations of it must be tested to find the best fit.

I. Introduction

Convolutional Neural Networks (CNN) are a subset of deep learning methods primarily used for image/video recognition, image classification, medical imaging analysis and other applications. In this analysis, a CNN was used and studied for the application of image classification using the CIFAR-10 dataset. The image classification problem is such that given an image, the CNN algorithm's objective is to predict the image class/type (e.g., car, animal, etc.) using a trained model. The CNN model is trained using a sufficiently large training dataset. Once the CNN model is trained, it can predict the classification using a testing dataset which the CNN model has not seen before. Two CNN architectures were considered for image classification –a plain CNN and one that was couple with a Residual Network (ResNet). Furthermore, analysis of both architectures was concentrated to areas where CNNs are known to have deficiencies, such as overfitting and vanishing gradients. Several CNN architectures parameter variations were studied and evaluated in terms of prediction performance. The CIFAR-10 dataset consists of 60,000 images for 10 classes with 6,000 images per class. The classes and sample images are shown in Figure 1. The dataset is divided into 50,000 images for training and 10,000 for testing.

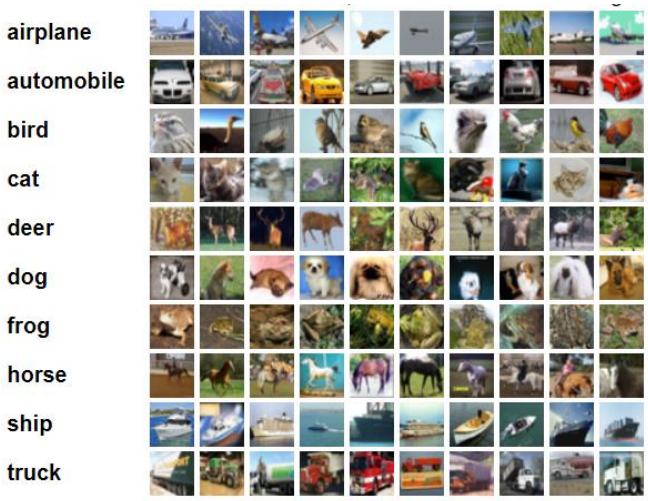

Figure 1: Illustration of CIFAR Image Dataset

II. Methods

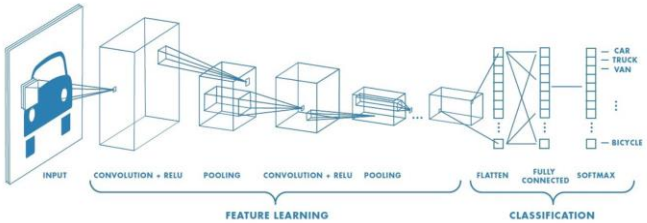

Figure 2: General CNN Architecture

A CNN is a type of deep learning algorithm that has a structure composed of a feature learning stage and classification stage, as shown Figure 2. In the feature learning stage ,the network may include multiple layers of convolutional and down-sampling combinations, while the classification stage makes use of the Multi-layer Perceptron

(MLP) to determine the class or label of the input data using the output of the feature learning stage. The convolutional layer makes use of filters, or kernels, to learn the features of the data by doing a convolution operation. This can be summarized as an element-wise product and sum between the input matrix and the filter matrix, where the output is then passed through a non-linear activation function. The convolution operation is illustrated in Figure 3.

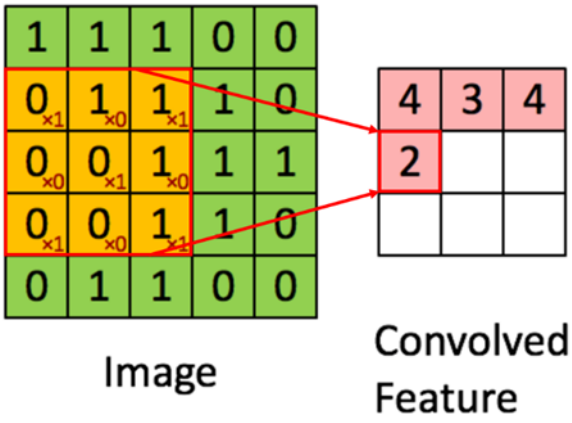

Figure 3: Convolution Illustration

The purpose of the activation function is to introduce non-linearity in the feature learning stage of the network to account for the fact that the input data (e.g., images) is normally non-linear. There are several types of activation functions available which include linear, sigmoid, tanh and rectified linear unit (ReLu). The most common activation function used in the convolution layer is the ReLu function which is shown in Figure 4.

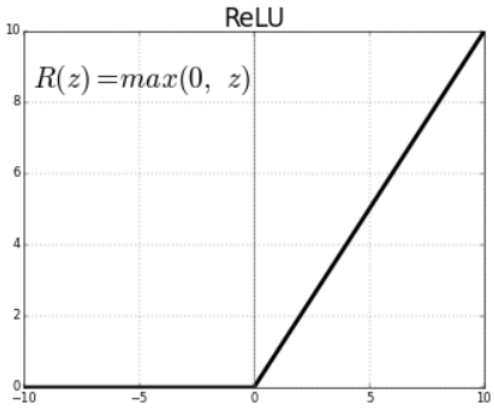

Figure 4: ReLu Activation Function

The down-sampling, also known as pooling, is applied after the convolutional layer in order to reduce the dimension of the data from the previous layer. The most common down-sampling method is Max Pooling, for CNNs, which uses the max value from each of a cluster of neurons at the prior layer [1]. The computation is illustrated in Figure 5. The result of max pooling is the maximum value over the area where the window is applied (denoted by the red box). The window moves across the input data in number of steps – known as strides. One of the benefits of using down sampling is that the network can focus on fewer neurons which has a regularizing effect on the network, which helps mitigate overfitting the training data.

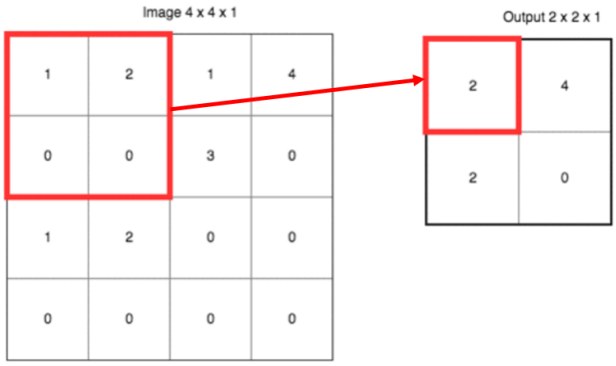

Figure 5: Max Pooling

In the classification stage the input data is flattened into a feature vector and passed through a Multi-Layer Perceptron (MLP), which is a neural network which consists of at least three layers –an input layer, hidden layer, and output layer. The output of the MLP is the prediction of the class of the data (e.g. images).

A. Convolutional Neural Network Architecture

The CNN architecture used in this analysis is shown in Figure 6. The architecture consists of three blocks for the feature learning stage and a MLP for the classification stage. Each block within the feature learning stage consists of a double sequence of layers consisting of convolution->normalization->max pooling->dropout with the exception of the first block where there is only one max pooling layer. The convolution filter sizes are 64, 128 and 256 for Block 1, Block2 and Block 3, respectively. A ReLu activation function was used for the convolution layers. Max pooling with a window size 2x2 and stride of 2 was used for the down-sampling layers within all blocks. Dropout layers were added throughout the network in order to protect against overfitting to the training data.

The classification stage uses an MLP with two dense layers and dropout is also used. The first dense layer of the MLP has 64 nodes and uses a ReLU activation function, while the last dense layer has 10 nodes (matches number of classes in data) and uses the Softmax activation function.

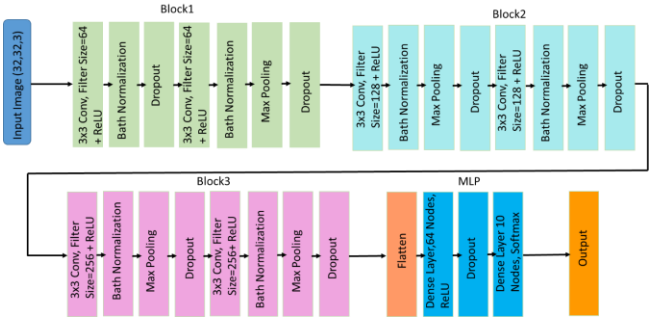

Figure 6: Architecture of CNN used for this analysis

Several areas of the CNN architecture, such as number of layers used, were investigated with respect to the classification performance.

B. *Convolutional Neural Network with ResNet*

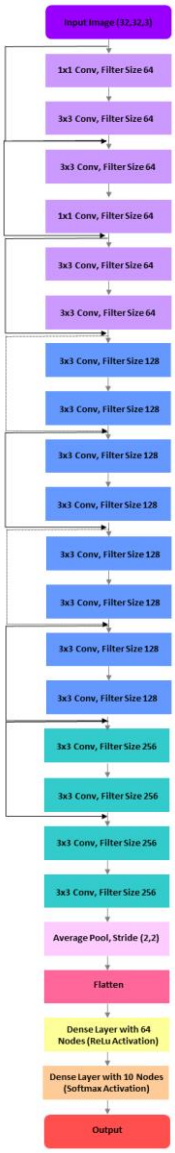

Figure 7: Architecture of CNN + ResNet

Residual Networks, referred to in this text as ResNet, were first introduced in 2015 to address the problem that arises when training Deep Neural Networks. The idea that adding more layers to your model would in turn increase its performance had been put to the test when a CNN with 20 layers was compared to one with 56 layers in the paper "Deep Residual Learning for Image Recognition" [3]. Not only did the model with additional layers result in larger training error on the dataset, but it also performed worse on the testing data. Therefore, this alluded that more complex models do not fail because of overfitting, but the ever present vanishing gradient problem. The general idea behind a CNN is that each subsequent layer is focused on learning more complex features of the image. However, when it comes time to calculate the predicted error and translate that to gradient that can be used to adjust the weights at each one, a deeper network may have a harder time getting that relevant information all the way through to the layers at the front end. That's when He, Zheng, Ren, and Sun introduced ResNet to combat the issue [3].

The basic idea behind ResNet is that you are taking the output from an earlier layer and adding it to a current layers output –also sometimes called a "skip" connection since it passes over some layers in your network. The motivation began with thinking that a shallower model should not perform worse than its more complex counterpart [2]. Therefore, it was suggested to feed information learned in earlier parts to later parts with the hopes of increasing depth in the network while not necessarily increasing computational complexity.

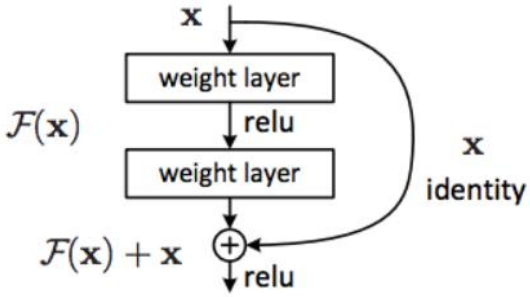

Figure 8: Example of a skip connection for the residual layer [4]

Figure 8 depicts the architecture for the residual connections. In the case where two layers with matching dimensions are added together, no additional computations are needed for the information being skipped forward. However, when they differ, a simple 1x1 convolutional computation can be performed on the skip connection with a filter size matching the dimension of the output it will be added to. In the CNN + ResNet model implemented in this paper, where skip connections were used, ReLu activation was applied after the addition, followed by batch normalization, dropout, and max pooing in some cases.

III. Analysis

A. *Analysis of Plain CNN*

One area of interest was to study the depth of the network in terms of training and testing performance. Three different networks were trained and evaluated, starting with one block in the feature learning stage and using up to three blocks. Training accuracy and loss are shown in Figure 9 and Figure 10, respectively. The training accuracy after 200 training rounds was 89.3%, 95.7% and 97.9% for 1 block, 2 blocks and 3 blocks respectively. Training accuracy does not necessary reflect the performance of the CNN algorithm, but it tells that it is able to learn the data

better as the number of layers increase.

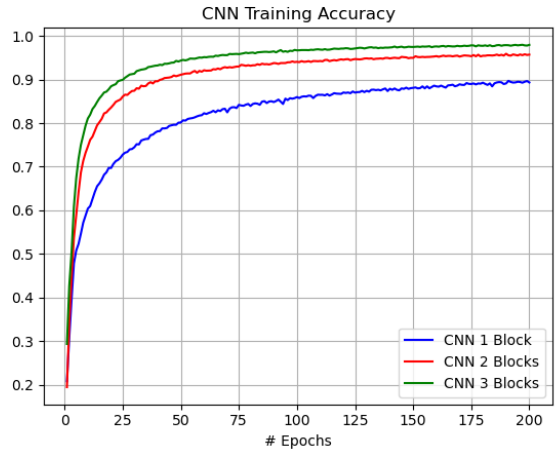

Figure 9: CNN Training Accuracy – Depth Comparison

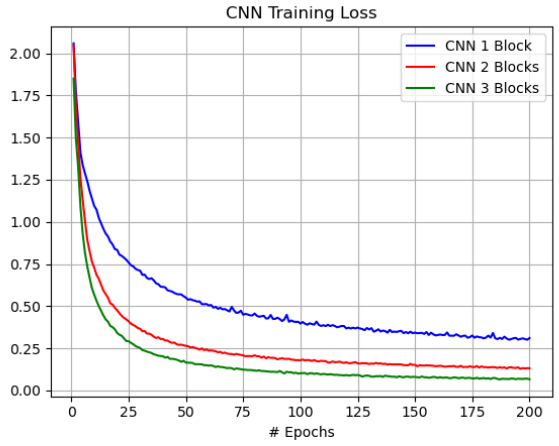

Figure 10: CNN Training Loss – Depth Comparison

Testing accuracy and loss are shown in Figures Figure 11 and Figure 12, respectively. The testing accuracy after 200 training rounds was 72.4%, 84.7% and 88.1% for 1 block, 2 Blocks and 3 blocks respectively. Results showed that using a deeper network, in this case 3 blocks, proved to have better training accuracy. The difference is mostly observed when adding a second block in the feature learning stage was added, where performance increase by 12% in testing accuracy.

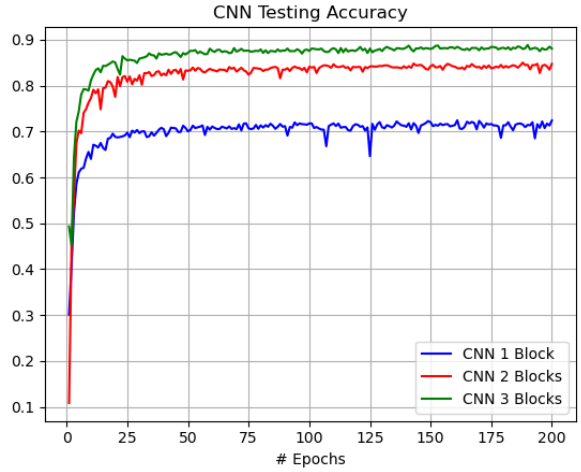

Figure 11: CNN Testing Accuracy – Depth Comparison

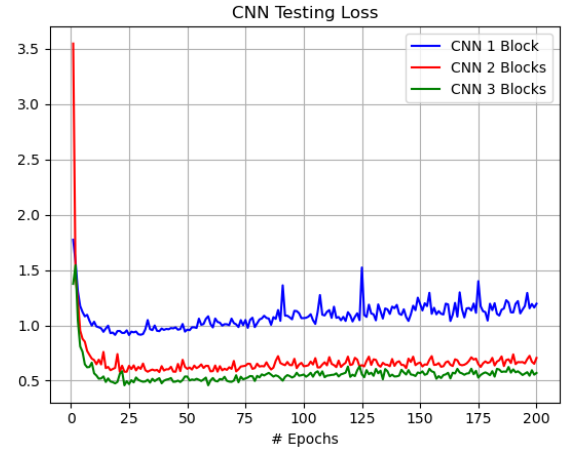

Figure 12: CNN Testing Loss – Depth Comparison

Another area of study was to investigate the impact of having dropout layers in the network. Deep learning algorithms are known to suffer from overfitting to the training data which results in a decrease in performance. This is specially a problem when the training data set is small, which is not the case in this analysis. Further, having a dropout layer is known to add a regularization effect on CNNs which in turn mitigate the overfitting problem [3].

Four CNN networks were compared to evaluate the impact of having no dropout at all and having dropout layer with different dropout rates. Results are shown in Figure 13 through Figure 16 for the training loss accuracy and testing loss/accuracy. Results show that having no dropout at all has higher training accuracy, but lower testing accuracy compared to any of the networks with having a dropout layer. More specifically, the training and testing accuracies with no dropout was observed to be 99.7% and 81.6% respectively. This indicated that the CNN was able to learn the training data better with no dropout but having a lower testing accuracy indicated the network suffered

from overfitting.

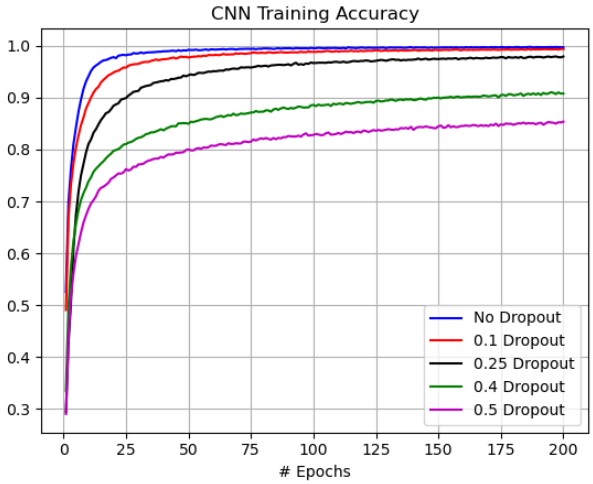

Figure 13: CNN Training Accuracy– Dropout Comparison

Furthermore, comparison of the CNNs with different dropout rates (0.1, 0.25, 0.4, 0.5) showed that increasing the dropout up to 0.25 showed better performance in terms of testing accuracy as observed in Figure 13. Increasing the dropout rate beyond 0.25 was observed to have worse testing accuracy across number of rounds. Note that with a dropout rate of 0.25 a larger testing loss was observed compared to 0.4 and 0.5 dropout rates yet it had better testing accuracy than 0.4 and 0.5 dropout rates. This was somewhat unexpected given that normally larger loss correlates with lower accuracy.

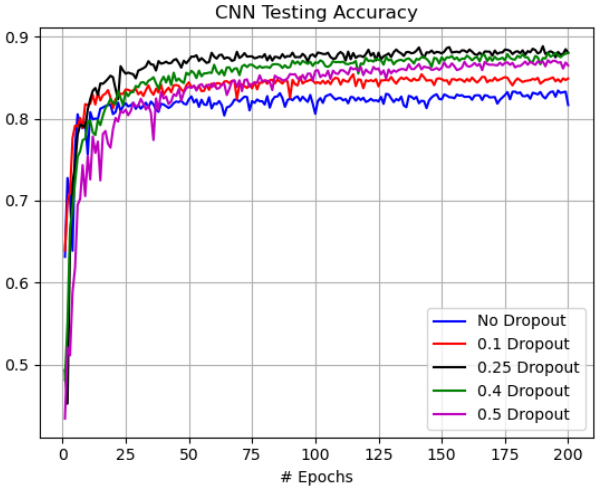

Figure 14: CNN Testing Accuracy – Dropout Comparison

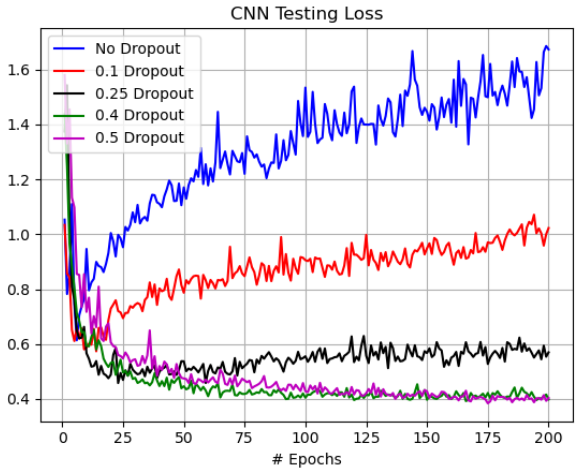

Figure 15: CNN Testing Loss – Dropout Comparison

B. *Analysis of CNN + ResNet*

To see the effects of using ResNet, the same architecture was tested by simply removing the skip connections. Training accuracy and loss are shown in Figures 16 and 17, respectively. The training accuracy after 200 epochs was 99.5% for the model using ResNet and 99.4% without. Using a CNN with ResNet did not produce the expected results yet, it leads to questioning whether the model was "deep" enough to yield them. Perhaps starting with a plain CNN with lower training accuracy would have been a better candidate to assess the benefit of adding ResNet. It is important to keep in mind that the addition of these skip connections is not to prevent overfitting on the training data, but to combat the vanishing gradient problem that plagues these types of deep neural networks. We expected the addition of skip connections to allow the network to achieve greater training and testing accuracy overall –as the idea was that the network itself was falling ill to the vanishing gradient problem. While both implementations performed well on the dataset, perhaps the addition of more layers would have been able to exploit the benefit of using ResNet.

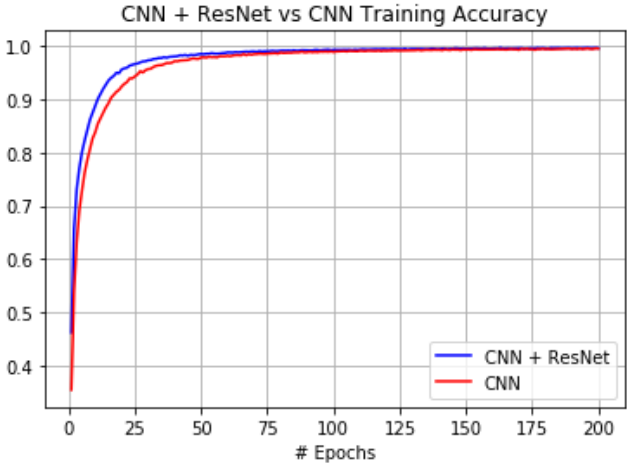

Figure 16: Training Accuracy – ResNet CNN

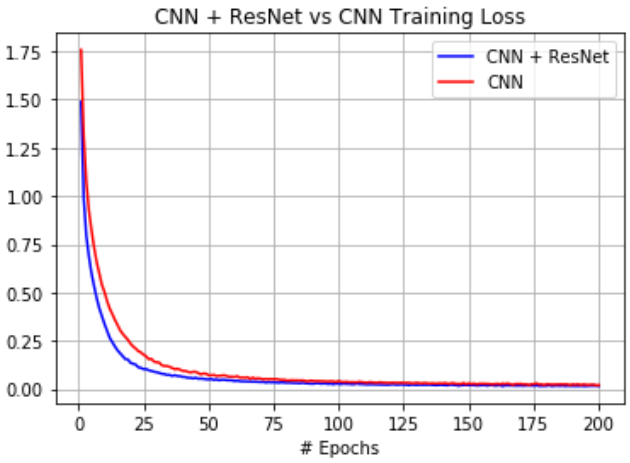

Figure 17: Training Loss – ResNet CNN

Since both models produced nearly identical metrics in terms of the training round, it was exciting to see a slightly more noticeable difference in performance during the testing round. Testing accuracies for both models are shown in Figures 18, and 19, respectively. Ultimately, the plain CNN resulted in a testing accuracy of 88.5% while the addition of ResNet achieved 86.5%. This may demonstrate some overfitting that occurred in the model that used ResNet, but perhaps this difference is not statistically significant. In order to draw that conclusion, more robust testing should be performed which could involve training the same configurations multiple times and reporting an average of their results.

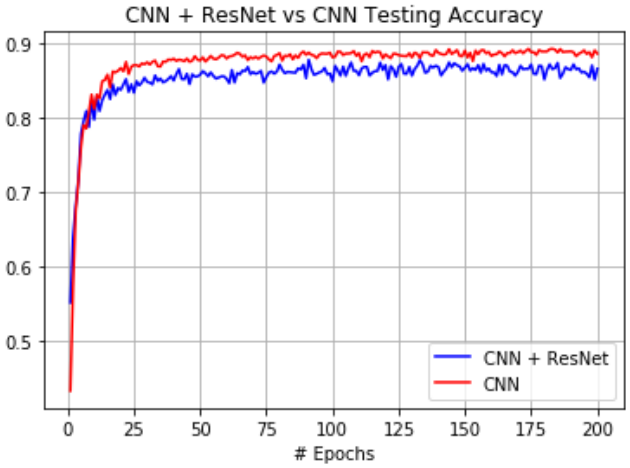

Figure 18: Testing Accuracy – ResNet CNN

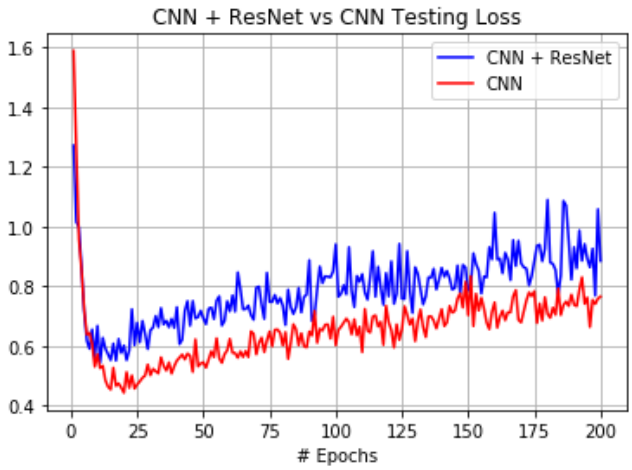

Figure 19: Testing Loss – ResNet CNN

The effects of adding dropout was discussed in detail in the analysis of the plain CNN, but the use of batch normalization can also act as a regularizer and make training the model more efficient. The figures below illustrate the effects of removing batch normalization from both models. In the previous example, batch normalization had been applied at every input to a layer.

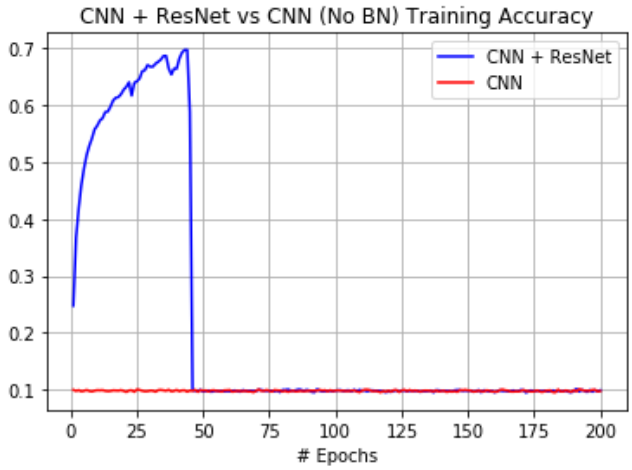

Figure 20: Training Accuracy –CNN + ResNet, no Batch Normalization

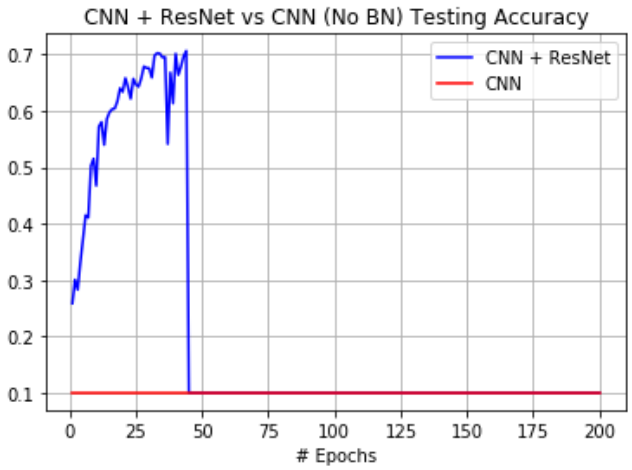

Figure 21: Testing Accuracy –CNN + ResNet, no Batch Normalization

There was a catastrophic drop in the training accuracy of the CNN + ResNet model when it reached epoch 46. This was most likely due to exploding gradients, since there was a dramatic change in the training metrics after that round and it could no longer produce reasonable weights to fit the data. Similarly, the plain CNN was unable to find any suitable weights and proved to be untrainable. When batch normalization is applied, it restructures the data so that it has zero mean and a standard deviation of one. This helps the model when backpropagation is applied, since it maintains some standardization between the layers and pushes for them to have similar distributions. The backpropagation algorithm relies on this assumption about distribution for its weight update procedure to be effective on the next training round. Removal of batch normalization made the model unstable, led to weights incapable of representing the data set features, and therefore unrecoverable. When it is used, it typically requires less epochs for the model to be trained effectively and clearly aids in stability. CNN + ResNet with batch normalization had a training accuracy of 98.18% at epoch 42, while the model without it reached its peak with a training accuracy of 69.1% before ultimately collapsing.

IV. Conclusion

Convolutional Neural Networks perform well due to their preservation of spatial information when propagating an input through the network, which is why they work so well for image recognition. One may think of these convolutional layers acting as a person with a flashlight looking for certain features. With each additional layer, they focus on more complex features to learn –from edges, to eyes, to sky, to ears. However, these networks still fall victim to overfitting, vanishing gradients, and even exploding gradients. There are some heuristics when it comes to determining the filter size, strides, and number of layers –but ultimately, these networks are not deterministic. Trial and error leads to exploration of what configuration will best determine the fit of your model. Similarly, adding more layers to make it deeper may also result in an inability for it to propagate useful information to shallower parts – for that, ResNet was explored. It's clear that by removing layers, dropout, and batch normalization we were able to see the effects of overfitting and vanishing/exploding gradients from an initial well performing model. Ultimately, multiple configurations of a model should be fit to see what will best learn according to your training data.

References

[1] Yamaguchi, Kouichi; Sakamoto, Kenji; Akabane, Toshio; Fujimoto, Yoshiji (November 1990). A Neural Network for Speaker-Independent Isolated Word Recognition. First International Conference on Spoken Language Processing (ICSLP 90). Kobe, Japan.

[2] He, Kaiming, et al. Deep Residual Learning for Image Recognition. Microsoft Research, 2016

[3] Ghahramani; BAYESIAN CONVOLUTIONAL NEURAL NETWORKS WITH BERNOULLI APPROXIMATE VARIATIONAL INFERENCEE.

[4] Shorten, Connor. "Introduction to ResNets." *Medium*, Towards Data Science, 15 May 2019