# CS-GY 6953 Residual Network Mini-Project

**Ping Chang, Jincheng Liang, Zhuo Xu**
https://github.com/ZhuoXu-CSE/Deep-Learning-Mini-Project

**Abstract**

Residual networks allow users to take the activation from one set of layers and feed it to the next set of layers to get much deeper into their neural networks. These architectures can be implemented with skip connections, or shortcuts, that contain batch normalization and ReLU layers in between. This allows users to easily optimize their neural networks while training error continues to decrease with an increase of training layers. With fewer layers to propagate through, the speed of learning becomes more efficient. This paper presents a short overview of our project on using a modified residual network architecture for the CIFAR-10 image classification dataset with a restriction of having less than 5 million parameters.

## Introduction

The CIFAR-10 image classification dataset consists of 60000 32x32 color images in 10 classes, 50000 of them being training images while the other 10000 are test images (Brownlee, 2019).

Residual network, also known as ResNet, is used to solve complex problems by stacking additional layers in deep neural networks to improve accuracy as well as performance (Yadav, 2022). Adding more layers allow more complex features to be progressively learned. However, there is a depth threshold with convolutional neural networks where at a certain point, adding more layers does not help with raising the performance.

This problem can be solved by introducing the use of residual blocks. These blocks create shortcuts, also known as skip connections, that allow for training on more shallow sub-networks. (Krizhevsky et. al. 2012). Without these connections, the input gets multiplied each time by the weights of the layer along with a bias term, resulting in long training times and not training deep networks effectively.

Figure 1 (He et. al. 2015) presents what a skip connection looks like, where in addition to a layer feeding into the next layer, it directs it into the layers that are further away.

This project aims to modify the ResNet architecture on the CIFAR-10 image classification dataset with the highest test accuracy possible provided that the developed model should not have more than 5 million parameters.

Our team attempted to use residual blocks to downsize the number of parameters, which allowed for sub-networks to be easier trained with these shorter paths. A 9-layer residual network was created by modifying from a larger ResNet, where we trained the architecture for 100 epochs to reach around a 91% accuracy. The details on how the model was designed will be presented in the methodology section as well as a results section that shows figures of the train and test accuracies.

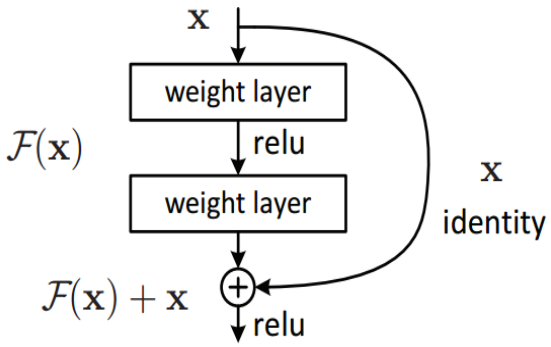


Figure 1: Skip Connection on ResNet

## Methodology

Over the past few weeks, our team has been investigating ways to modify the residual network architecture under the constraints of not having more than 5 million parameters while still retaining a decent performance on the CIFAR-10 image classification set. Since ResNet-50 has more than 25.6 million parameters and even ResNet-18 has around 12 million trainable parameters, we decided to explore some smaller-scale models. As ResNet-18 was able to achieve a 93.02% test accuracy, we decided to use this model as a base and attempted to configure it to develop an architecture with less than the maximum trainable parameters allowed and with minimal performance degradation.

Residual blocks were effective in allowing for a more efficient training time as well as reach a higher accuracy. Our team agreed on using the code for the residual block as defined by He et al. (He et. al. 2015)

The residual blocks contain an identity shortcut and preserve the spatial and channel dimensions of the input. In contrast, convolutional residual blocks reduce the spatial

resolution by a factor of two and double the number of output channels. The motivation for including these residual blocks is to simplify optimization where longer paths are to add computational depth instead.

One drawback we observed with this backbone network is that the down sampled convolutions have a kernel of 1 x 1 and a stride of 2. Instead of enlarging the receptive field, they discard the information (Myrtle.ai, 2022). If these convolutions were replaced with 3 x 3 convolutions of stride 1, the performance should be improved. We further enhanced the down sampling by adding a max pooling layer with a 2 x 2 window size. The final pooling layer before the classifier is a concatenation of the global average pooling and max pooling layers. Finally, residual branches were added after the first and third residual blocks.

Our team settled on using the Adam optimizer as through testing, it required less parameters for tuning and a faster computational time than SGD. Although there are some known disadvantages with using Adam, such as problems with weight decay, we have not run into those but will continue to monitor our network in case these issues come up.

To confirm our results made sense, the network was trained with both a batch size of 128 and 256, where other hyperparameters were kept the same. Our team wanted to test whether a larger batch size will affect the test accuracy in any way, whether the quality of our model will degrade with a larger batch that allows the network to be ran more quickly. Additionally, the learning rate was set to 0.001. Our team felt that setting the learning rate any lower than the above value does not provide a better training result and the time it took for optimization was drastically longer. Finally, the weight decay was set to 1e-5 as this ratio with the learning rate seemed most like that of ResNet-50 and ResNet-18.

Even though the test accuracy curve reaches its peak at around 30 epochs, we decided to run it for 100 epochs to see if the residual blocks were working as intended and that there was no sudden accuracy drop off.

## Results

Our final model is a residual network with 9 layers. Each 2D convolution layer is followed by a 2D batch normalizer and then a ReLU function. This satisfied the condition of having less than 5 million parameters, where our model has around 2.4 million parameters as shown in Figure 2. Figure 3 is a visual representation of what the developed residual network look like.

Residual blocks were utilized to test deep neural networks and prevent degradation from affecting the performance of model training.

```
================================================================
Total params: 2,447,946
Trainable params: 2,447,946
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 8.76
Params size (MB): 9.34
Estimated Total Size (MB): 18.11
----------------------------------------------------------------
```

Figure 2: Total number of Parameters

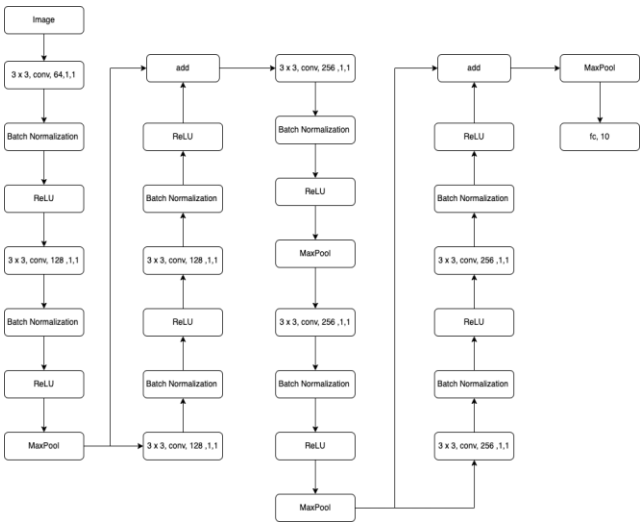


Figure 3: ResNet-9 Model Architecture

After training the architecture for 100 epochs, the training accuracy closely approaches 100%. Both results from a batch size of 128 and that of 256 had around a 91% test accuracy with an average loss of less than 1. The result from a batch size of 128 had a slightly less loss by 0.01, which is negligible and proves that a larger learning rate does not degrade the quality of training. Figures 4 and 5 are graphs that presents the accuracy through the number of epochs tested for the batch size of 128 and 256 respectively.

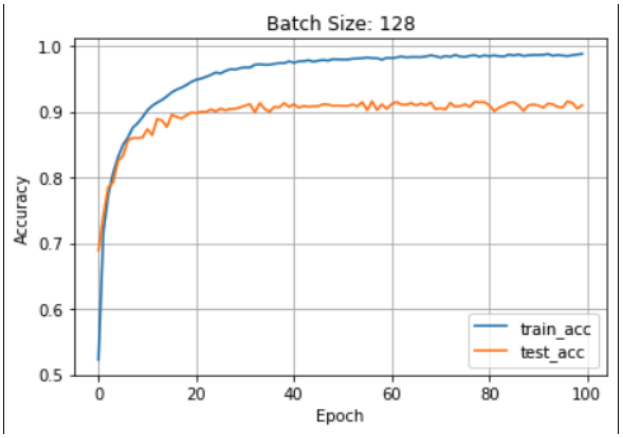


Figure 4: Accuracy vs Epoch for 128 Batch Size

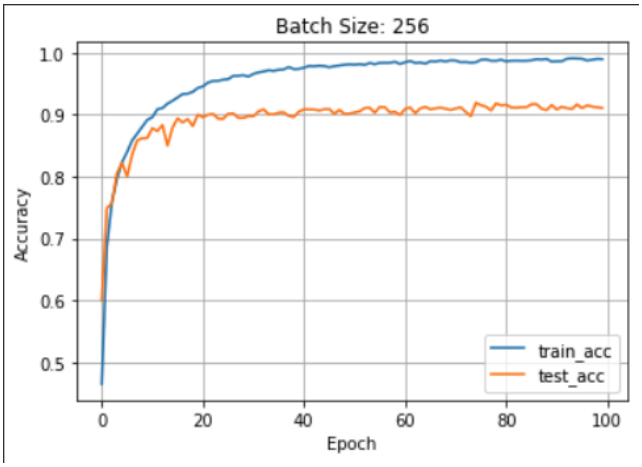

Figure 5: Accuracy vs Epoch for 256 Batch Size

Although there are slight dips in the training accuracy throughout the progression, it is only a 0.05 loss at most. This proves that the residual blocks placed within the architecture is effective as without it, the training accuracy will drop by a lot more.

Our team has successfully created a residual network utilizing 9 layers that had a total number of parameters of around 2.4 million. We were able to reach the minimum baseline of 80% test accuracy and more. The source code can be found at the GitHub link provided on the first page of this report.

## References

**Deep Residual Learning for Image Recognition**
He K.; Zhang X.; Ren S.; Sun J. 2015. Deep Residual Learning

**How to Train Your ResNet 4**
Myrtle.ai. 2022. Architecture. https://myrtle.ai/learn/how-to-train-your-resnet-4-architecture/. Accessed: 2022-11-21.

**ImageNet Classification with Deep Convolutional Neural Networks**
Krizhevsky A.; Sutskever I.; Hinton G. 2012. The Architecture.

**Machine Learning Mastery**
Brownlee, J. 2019. How to Develop a CNN From Scratch for CIFAR-10 Photo Classification. https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/. Accessed: 2022-11-20.

**Residual Blocks in Deep Learning**
Yadav H. 2022. Residual block, first introduced in the ResNet paper solves the neural network degradation problem. Towards Data Science.