Implementation of Resnet for CIFAR-10

Yoonjeong Park
Department of Computer Science
Yonsei University
dbw2140@yosei.ac.kr

Abstract

The original paper "Deep Residual Learning for Image Recognition" has made it easier to train deep models and has had a significant impact on the field of deep learning, as evidenced by its high number of citations. ResNet, in particular, has been used as a backbone architecture in many deep models.
However, most ResNet implementations from scratch have followed the architecture defined for ImageNet. According to the original paper, the structure of ResNet for ImageNet and ResNet for CIFAR-10 are different. In fact, when searching for ResNet implementations for CIFAR-10, it can be observed that there are less code for it than the one for ImageNet.
Given that the performance of an individual GPU is limited, I attempted to investigate whether residual learning effectively addresses the degradation problem by implementing ResNet for CIFAR-10 with smaller image resolutions and datasets instead of ImageNet. I followed the structure specified in the paper and found that, unlike the plain 56-layer net, the training loss of the ResNet56 model decreases consistently during training, indicating that it effectively addresses the degradation problem that arises with increased layer depth.

1 Introduction

According to the original paper, the depth of a neural network plays a crucial role in determining its performance, and deeper models tend to perform better. However, the original paper also raised the question, "Is learning better networks as easy as stacking more layers?"

The issue of gradient vanishing in deep networks has been largely addressed through techniques such as normalized initialization and batch normalization. Although deeper networks have become more convergent, the original paper points out that as the depth of the network increases, another problem arises where accuracy saturates and quickly degrades. This degradation problem is evident from the higher training error observed in deep models with more layers. In other words, this degradation problem suggests that not all systems optimize in a similar manner.

Nevertheless, the original paper proposes a way to optimize deeper models in limited situations. By adding an identity mapping layer and copying the other layers from a shallower model, the deep model should have a training error that is not larger than the shallower model. The authors of the paper applied this concept to their model in a restricted setting.

To address the degradation problem, the original paper introduced a "deep residual learning framework" which allows stacked layers to directly fit the underlying mapping they aim to learn, rather than fitting a residual mapping.

In Plain networks, the goal is to obtain a function H(x) that maps input x to the target output y. In contrast, ResNet modifies this goal to obtain H(x) - x, which represents the difference between the output and input.

Preprint. Under review.

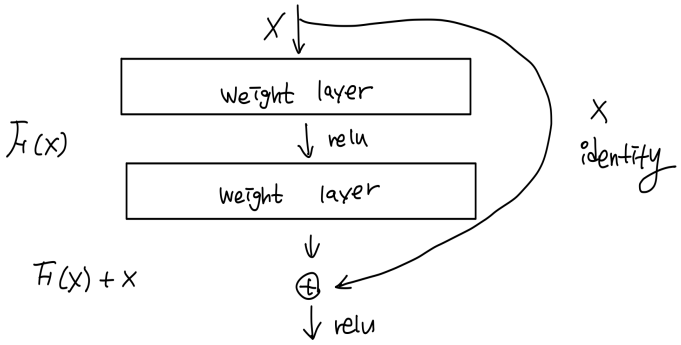

Figure 1: residual learning block

Consequently, the residual function F(x) = H(x) - x should be minimized, where x represents the output of the previous layer and cannot be changed along the way. Therefore, the optimal solution for F(x) is for it to become 0, which ultimately makes the learning goal to map H(x) to x.

The authors of the paper argue that learning the reserved mapping (F(x) = H(x) - x) is easier than learning the unreferenced mapping (H(x)) because the optimal target value of H(x) = x is provided as a pre-conditioning.

In ResNet, the original mapping H(x) = F(x) + x is implemented through "shortcut connections" in the feedforward network. These shortcut connections, which skip one or more layers, perform identity mapping in the case of the original paper. The outputs of the shortcut connections (x in figure1) are added to the outputs of the stacked layers (F(x) in figure1).
These identity shortcut connections do not introduce extra parameters or computational complexity(in case of OptionA), making the entire ResNet network easy to learn and implement end-to-end through backpropagation in the SGD method.

The original paper used a slightly different architecture of ResNet for ImageNet and CIFAR-10 datasets.
It has been shown to be helpful for deep neural network (NN) learning. Due to limitations in GPU resources, I implemented ResNet20, and Resnet56 for the smaller-sized CIFAR-10 dataset, following the approach of the original paper. Similar to the findings of the original paper,
my implementation of residual learning was shown to be effective in solving the degradation problem in experiments which will be discussed in Experiments Section.

2 Model/Method

This project implements five models: ResNet56 using OptionA and OptionB, ResNet20 using OptionA, Plain56-layer Net, and Plain20-layer Net, based on the original ResNet paper. The architecture of Plain Network and ResNet is the same as shown in the figure2,3. Also, below is the rules that model architecture follows.

• The input to the network is 32×32 images with the per-pixel mean subtracted.

• The first layer consists of 3×3 convolutions.

• The network then includes a stack of 6n layers with 3×3 convolutions, where n depends on the model (n=9 for ResNet56 and n=3 for ResNet20). These layers operate on feature maps of sizes 32, 16, 8, with 2n layers for each feature map size.

• The number of filters in each layer is 16, 32, 64 respectively.

• Subsampling is performed by convolutions with a stride of 2, reducing the spatial dimensions of the feature maps.

• Using Batch Normalization right after Convolution layers and right before ReLU activation functions.

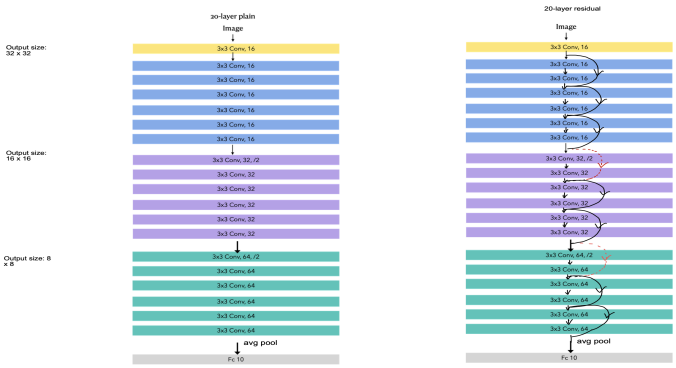

Figure 2: Architecture of plain20-layer network(left), resnet20(right)

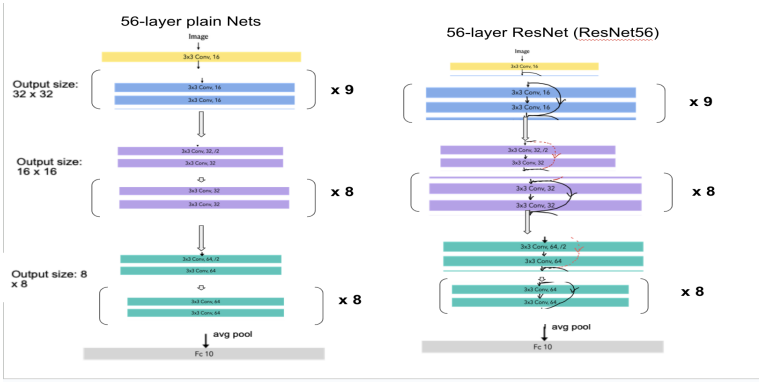

Figure 3: Architecture of plain56-layer network(left), resnet56(right)

• The network ends with global average pooling, a 10-way fully-connected layer, and softmax activation for classification.

• Overall, there are 6n+2 stacked layers with learnable weights in the network.

In the case of ResNet, a total of 3n shortcut connections are used, connecting pairs of 3x3 layers. In the Resnet structure of figure2, 3, the round arrow lines are inserted short connections. The identity shortcuts(figure 1) can be directly used when the input and output are of same dimension (black solid line in Figure 2, 3). When the dimensions increase(dotted red line shortcuts in Figure2, 3), the original paper consider two options: (A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions (you can see the code for it at the “56 resnet56 optionA" file in “code" folder). (B) The projection shortcut is used to match dimensions (done by 1x1 convolutions) (you can see the code for it at the “56 resnet56 optionB" file in the “code" folder).

In the original paper, identity shortcuts (option A) are used in all cases, where the input is added directly to the output of the stacked layers. The projection layer (option B: use 2 more conv layer as the projection layer to match the dimension) is not used. Therefore, residual models using optionA have exactly the same depth, width, and number of parameters as the plain counterparts.

In this project, I implemented ResNet20 using OptionA, ResNet56 using both OptionA and OptionB and the performance will be discussed in the Experiments Section.

3 Experiments

Resnet56, Resnet20, Plain 20-layer net, Plain 56-layer net were trained and I did a experiment to show the effectiveness of residual learning and compare which option is better to achieve better performance.

Model	# layers	#params	test error(%)
ResNet20 (OptionA)	20(optionA)	0.27M (269,722-OptionA)	0.279 = 27.9% (90.pth)
ResNet56 (OptionA)	56(optionA)	0.85M (853,018-OptionA)	0.285 = 28.5% (90.pth)
ResNet56 (OptionB)	56 + 2	0.85M (855,770-OptionB)	0.259 = 25.9% (90.pth)
plain20	20	0.27M (269,722)	0.325
plain56	56	0.85M (853,018)	0.326

Figure 4: number of parameters of Resnets, Plain nets and test errors

3.1 Training Method and CIFAR-10 dataset

For training, a weight decay of 0.0001 and momentum of 0.9 were used just like the original paper. These models are trained with a mini- batch size of 128 on 1 GPU of google Colab. Training starts with a learning rate of 0.1, divide it by 10 at 32k and 48k iterations, and terminate training at 64k iterations, which is determined on a 45k/5k train/val split. I applied simple data augmentation for training: 4 pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip and normalize per-pixel with mean, std values. For testing, I only evaluate the single view of the original 32×32 image.

The CIFAR-10 dataset is a widely used benchmark dataset in machine learning and computer vision research. It consists of 60,000 color images, each with a resolution of 32x32 pixels, and is divided into 10 classes, with 6,000 images per class. The dataset is further split into a training set of 50,000 images and a test set of 10,000 images.

The CIFAR-10 dataset is commonly used for tasks such as image classification, object recognition, and image synthesis. It is often used as a benchmark dataset for evaluating the performance of machine learning models and comparing their accuracy and generalization ability.

3.2 Comparison of Training Error between residual nets and plain nets

When the performance was evaluated on a 10K testdatset after learning by 163 epochs with a 50K trendatset of CIFAR-10, it was as follow. See figure 4.

See figure5 and figure6. In figure5, the training loss of plain 56-layer net is bigger than the one of plain 20-layer net. However, in figure6, the training loss of residual 56-layer net is smaller than the one of residual 20-layer net. Therefore, it is shown that the residual learning is effective to solve the degradation problem of stacking more layers and crucial for deeper networks.

3.3 Comparison of Training Error between Residual mapping using OptionA and OptioinB

In the original paper, the authors mention that Residual mapping using OptionB is slightly better than the one using OptionA. However, they just use all residual mapping of model as OptionA for CIFAR-10 dataset. I implemented both resnet56 with OptionA residual mapping and OptionB and compare their training loss and test error. Like the original paper said, the performance of OptionB is slightly better than OptionA. (see figure 7)

4 Conclusion Future Work

In this project, ResNet20 and ResNet56 models were implemented for the CIFAR-10 dataset, and the experiments showed that residual learning effectively addressed the degradation problem in deeper networks.

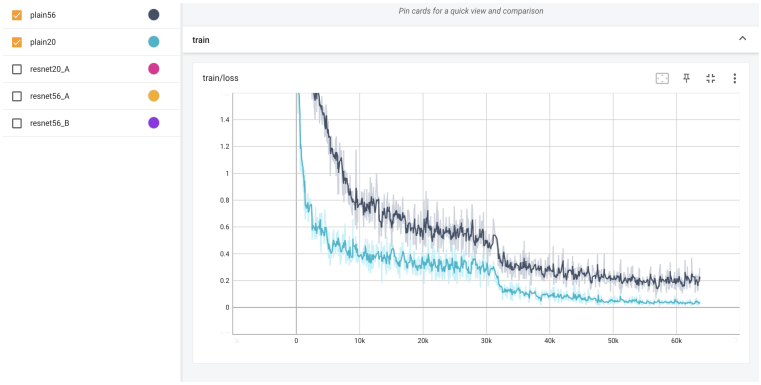

Figure 5: train loss of Plain nets. Blue line is train loss of 20 layer network. Black line is train loss of l56 layer network

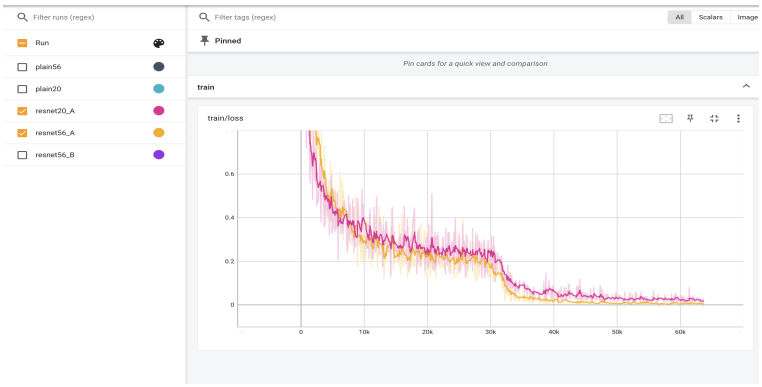

Figure 6: train loss of Residual nets. Pink line is train loss of ResNet20. Yellow line is train loss of ResNet56

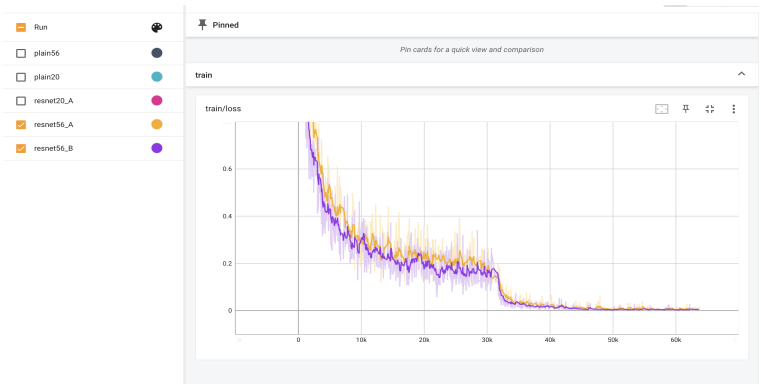

Figure 7: Training loss of OptionA, OptionB. Purple line is Resnet56 with OptionB. Yellow line is Resnet56 with OptionA

Although the degradation problem was solved by reducing training error, finding the best model through changes in learning methods such as learning rate and epochs was challenging. As a result, the test error was not as low as reported in the original paper.

Similar to the approach in the original paper, adjusting hyperparameters such as learning rate and epochs using a validation dataset to select the best model is expected to improve the performance in terms of test error. This could be a potential future direction for further research and improvement in this project.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.