Deep Learning Mini Project

Juquan Yu[1], Shuai Zhang[2], Ruichen Yang[3]

Fall 2022, NYU Tandon School of Engineering
jy3873@nyu.edu[1], sz3714@nyu.edu[2], ruichen.yang@nyu.edu[3]

Abstract

For this mini project, we followed the basic conception of the original ResNet architecture as our blueprint and modified it to meet the given requirements. Finally, we built the network with 4,985,386 trainable parameters, then got 90% test accuracy in the CIFAR-10 Dataset with 40 epochs. The codebase of the project is here. https://github.com/KurisuZhang/Deep-Learning-Mini-Project/blob/main/README.md

Introduction

ResNet

ResNet [2] is developed to alleviate problems of training deep networks. The most essential modification is the introduction of *Skip Connection*. To be specific, ResNet adds the output of the layer before the previous layer directly to the next layer, which will eliminate the problem of vanishing gradients. See Figure 1 demonstrates a residual block inside a ResNet.

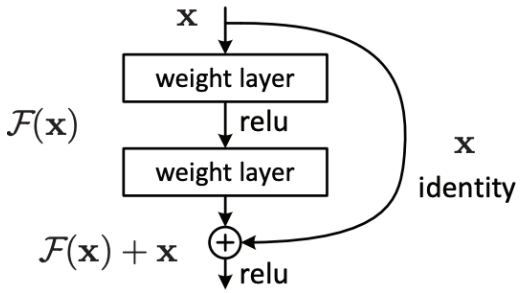

Figure 1: A residual block [2]

Dataset

We used CIFAR-10 as our dataset. This dataset is collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton [3], which containing $60,000$ colored images with 32×32 resolution in 10 different classes. There are 50,000 images for training and 10,000 images for testing. The 10 classes are bird, airplane, ship, dog, deer, frog, truck, horse, cat, and automobile. The classes with each other are completely mutually exclusive. Figure 2 shows some example images with corresponding labels in this dataset.

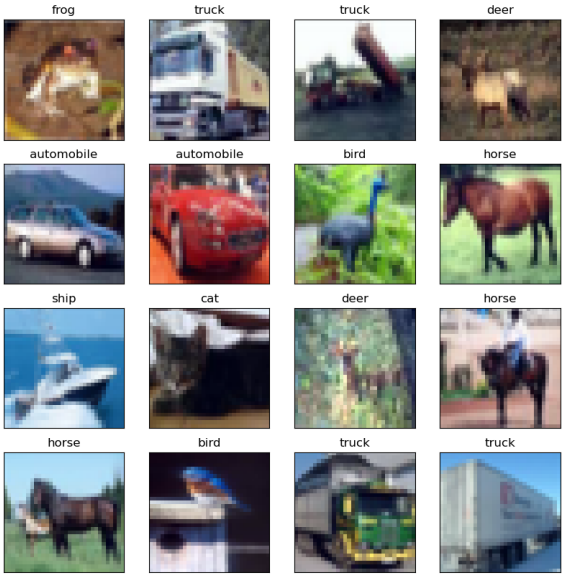

Figure 2: 16 random selected images in CIFAR-10

Network

We revised the architecture of the original ResNet-18 to deliver a better performance on CIFAR-10 image classification dataset but limited it to less than 5 million trainable parameters.

Architecture

We deepen the original ResNet [2] by reducing the number of channels, allowing to add more layers and achieve higher accuracy with fewer parameters.

We customize a residual block with six layers; the skip connection is added from the first layer to the middle of the five and six layers:

- Layer 1. Conv2d
- Layer 2. BatchNorm2d
- Layer 3. ReLU
- Layer 4. Conv2d
- Layer 5. BatchNorm2d
- Skip Connection from Layer 1

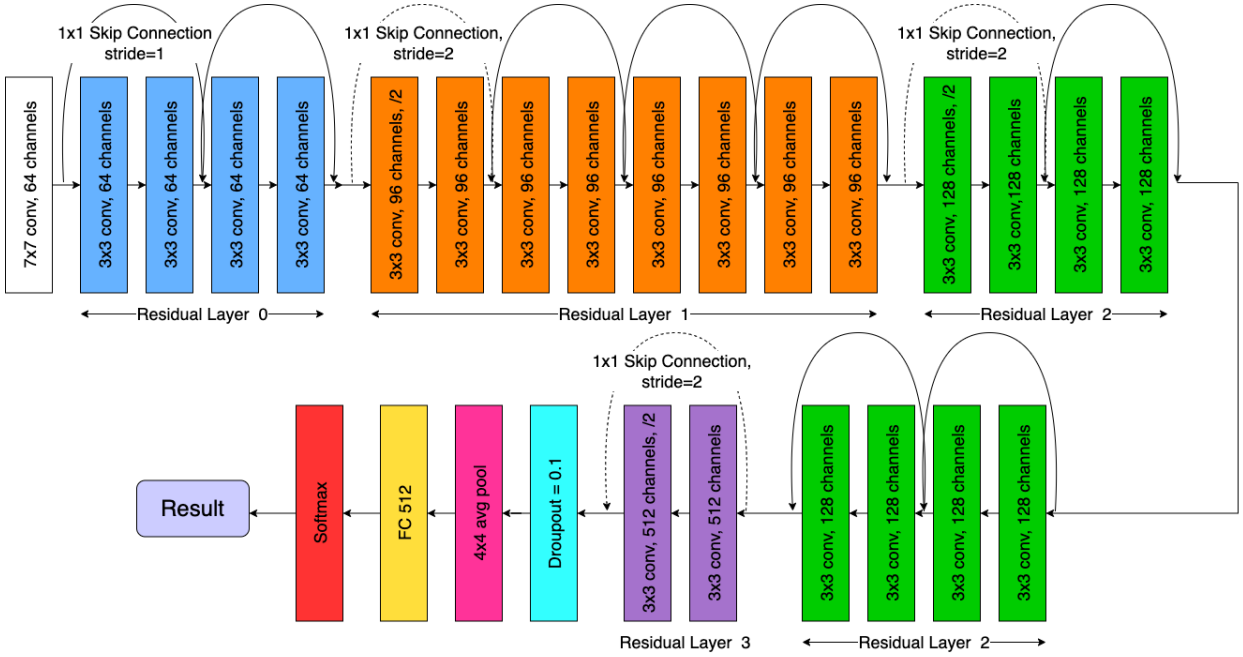

Figure 3: Architecture of our network

- Layer 3. ReLU

The first part is 7×7 convolution layer; after this, the shape of the network becomes [64, 32, 32]. The second part customized residual layer 0, which is a customized 64 channel 3×3 residual block, repeat for twice; after this, the shape of the network becomes [64, 32, 32]. The third part customized residual layer 1, which is a 96 channel 3×3 residual block, repeat four times; after this, the shape of the network becomes [96, 16, 16]. The fourth part customized residual layer 2, which is a 128 channel 3×3 residual block, repeat four times; after this, the shape of the network becomes [128, 8, 8]. The fifth part customized residual layer 3, which is a 512 channel 3×3 residual block, repeat four times; after this, the shape of the network becomes [512, 4, 4]. The sixth part is to add a dropout layer, 4×4 average pooling layer, and a fully connected layer, as shown in Figure 3. The total trainable parameters are 4,985,386.

Parameters. In the above network, the following parameters are used in our model:

- **B**: $[2, 4, 4, 1]$: this is the number customized of residual blocks in each customized residual layer.
- **C**: $[64, 96, 128, 512]$: this is the number of channels in each customized residual layer.
- **Stride**: $[1, 2, 2, 2]$: this is the stride of the first convolution layer in our customized residual layer.

Optimizer

For the optimizer, we chose Stochastic gradient descent (SGD). The learning rate is a crucial parameter for SGD. The algorithm will go through many iterations to converge, but if the learning rate is too high, we may continue jumping over the ideal value.

Parameters. The parameter of the optimizer that we chose are shown below:

- **LR**: 0.05. This is the initial learning rate of our optimizer.
- **Weight_decay**: $1e-5$. This is used for regularization.
- **Momentum**: 0.9. This is the momentum value.

Pros & Cons

We mainly made the following adjustments on the basis of ResNet-18. During the training process, we halved the number of channels of each layer based on the original model, and appropriately reduced the number of blocks in several layers, so that our training parameters were controlled within 5 million. Although such a design reduces the accuracy of the model to a certain extent, it reduces the complexity of the model and improves the training speed.

As for the optimizer, we selected the SGD optimizer. Because it randomly selects a batch from the training set samples to calculate the gradient once and update the model parameters once, its training speed for large datasets is very fast. The disadvantage is that its learning rate parameters are difficult to adjust, and it is easy to converge to a local optimal solution. After several attempts, we adjusted the learning rate from the initial 0.1 to 0.05, so that the model finally reached a prediction accuracy of more than 90%.

In addition, we also added a Dropout layer. Dropout uses the idea of averaging to effectively reduce the complex relationship between neurons, thereby preventing over-fitting problems. Before adding the Dropout layer, the prediction accuracy of our model on the test dataset was always difficult to break through 90%, fluctuating around 88%. After adding the Dropout layer, although it increases the overall

training time, it effectively avoids the problem of the prediction rate drop caused by overfitting.

Improvements

We also use some tricks to prevent over-fitting and increase the accuracy of the test dataset; They are regularization, data augmentation, and dropout.

Regularization.

Regularization is a set of methods that can prevent overfitting in neural networks. Performing regularization will make the weight values not become too big [1]. As shown in the Equation 1, the loss function is not only to minimize MSE ($MSE(y, \hat{y}; \theta)$), but also the sum of weight ($\alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$).

$$J(\theta) = MSE(y, \hat{y}; \theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2 \tag{1}$$

Data Augmentation.

The amount and variety of data provided during training strongly affects the accuracy with which a supervised deep learning model can make predictions. Simply said, the number of learnable parameters in the model determines how much data is needed [6]. So, in our model, we try to use the below method to enrich the CIFAR-10 dataset: random horizontal flip, random crop, random rotation, and random brightness.

Dropout.

When a complex feed-forward neural network is trained on a small dataset, it can easily cause the overfitting problem. To prevent overfitting, the performance of the neural network can be improved by putting some feature detectors into the next layer [7]. In our model, we set dropout as 0.1.

Result

We write our code by using PyTorch framework, An open-source machine learning framework that shortens the time it takes to go from research prototype to production deployment [5]. We used kuangliu's GitHub code repository [4] as a reference and modified the code. The code runs on the cloud virtual machine of Google Cloud, with one Tesla P4 GPU. Codebase here: https://github.com/KurisuZhang/Deep-Learning-Mini-Project/blob/main/README.md

We have run our model total of 40 epochs and gained the best test accuracy of 91.26% in the No.36 epoch. The model nearly converges in 20 epochs. In the epoch between 30 to 40, the test accuracy basically stopped increasing and continued to oscillate around 90% The loss and accuracy figures are shown below. See Figure 4 & 5.

Specific to each class, the automobile class has the best prediction accuracy, which is up to 95.66%, but the cat class achieves worse performance, only 82.16% test accuracy. There are 8 classes that achieve over 90% accuracy. detail information is shown in Table 1 and Figure 6.

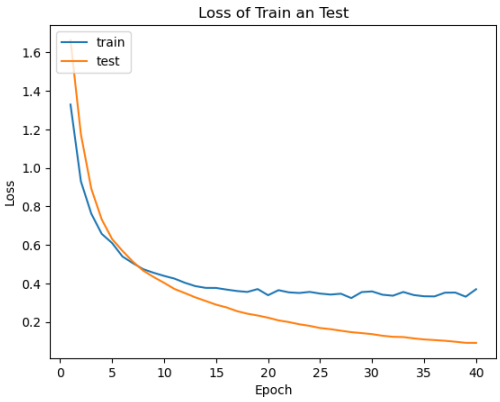

Figure 4: Train and test loss during 40 epochs

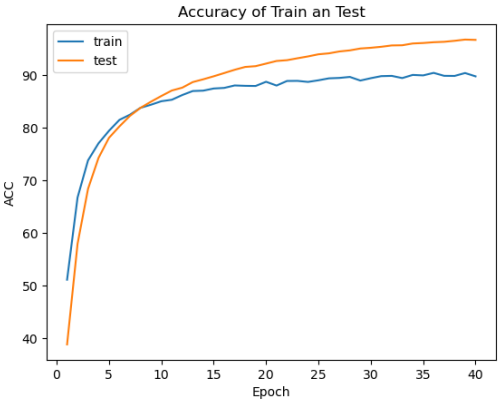

Figure 5: Train and test accuracy during 40 epochs

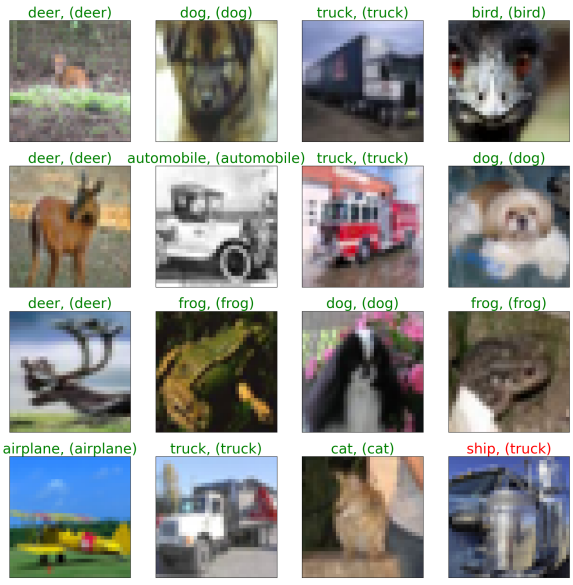

Figure 6: Prediction label and true label

Summary

We bootstrap our network with the existing ResNet-18 model. We find that the original ResNet contains 11,173,962

name of classes	Accuracy
airplane	93.462 %
automobile	95.661 %
bird	85.762 %
cat	82.1619 %
deer	92.862 %
dog	85.562 %
frog	93.862 %
horse	90.262 %
ship	93.562 %
truck	94.362 %

Table 1: Test accuracy of each class

parameters. After we reduced the channel size, extend the network depth, and used some optimization methods including regularization, data augmentation and dropout technique, the model ended up achieving over 90% accuracy with less than 5 million parameters.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Regularization for deep learning. *Deep learning*, pages 216–261, 2016.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[4] Kuang Liu. Kuangliu/pytorch-cifar: 95.47% on cifar10 with pytorch, https://github.com/kuangliu/pytorch-cifa.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[6] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.