CIFAR10 Image Classification

Zixuan Zhang

1 Introduction

Image Classification has has been one of the the hottest topics in machine learning. The immense math and experimenet behind this topic has led to some great discoveries and helpful researches.

In this background MNIST datasets[1] has been one fundamental and basic datasets for experiment as its input iamge only consist values for black and white, with a small dimension of 28x28 The next popular dataset of this topic would be CIFAR10 dataset[2].

The CIFAR10 dataset set consists of 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. They are RGB images and are bigger than CIFAR10 images, which means the input size is 3x32x32.

Using a traditional fully connected neural network would be sub-optimal in this case. One reason would be that the images are not stationary in any more, they vary in sizes colors. Fully connected neural nets would have a big problem with this kind classfication as it counts on every pixel of a image and trys to learn a parameter for that pixel. Convolutional neural networks would just be a much superior method in this case as it uses convolutional layers instead of fully connected layers.

[1] `http://yann.lecun.com/exdb/mnist/`

[2] `https://www.cs.toronto.edu/~kriz/cifar.html`

2 Methodology

There are many convolutional neural net structures that has been proven to be getting very useful over the years. LeNet, AlexNet, VGG[3], GoogleNet, Inception, and ResNet[4] were all some competiton winning structures that was the best at its time.

For this project, Pytorch[5] libary was used. Pytorch is an optimial libray for machine learning in general. Pytorch also provides a guide on how to train a CNN using the CIFAR10 dataset[6].

After reading many articles, it came to the decision of to use features of ResNet(Figure 1). My implementation consisted of a feature in ResNet called skip connections. In Figure 1, "bn_conv1" is concatenated with "conv3_relu" with the help of the "skip" layer, which is a layer that down-samples bn_conv1 to help match with the dimensions. Skipping connections can help with the problem of vanishing gradient that often occurs to CNN with very deep layers, the problem refers to when doing backpropagation, it would cause the gradient at the starting layers to have very small gradients essentially causing the parameters to underflow and equal to 0.

Batch normalization and ReLu layers were also used after every convolutional layer. It is a great aspect of the ResNet structure. Batch normalization helps layers to be more stable by re-center and rescaling, essentially being able to recognize images that are not always at the same positions. ReLu is a commonly used activation function that performs very well and also prevents the vanishing gradient problem.

Cross entropy loss is used for the natural of it being good for calculating loss with classification task of more than 2 classes. Adam optimizer was used for optimization as it is faster than SGD. The hyperparameters were tweaked many times for the purpose of faster convergence and higher accuracy, with the them being: learning rate of 0.01; 70 epoches; batch size of 64, weight_decay of 0.003.

When trying to get a higher accuracy, experiments with learning rate decay give a good result. Starting with 0.01, by multiplying 0.1 to the learning rate every 15 epoch, the model was able to converge better.

[3] `https://arxiv.org/abs/1409.1556`
[4] `https://arxiv.org/abs/1512.03385`
[5] `https://pytorch.org/`
[6] `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`

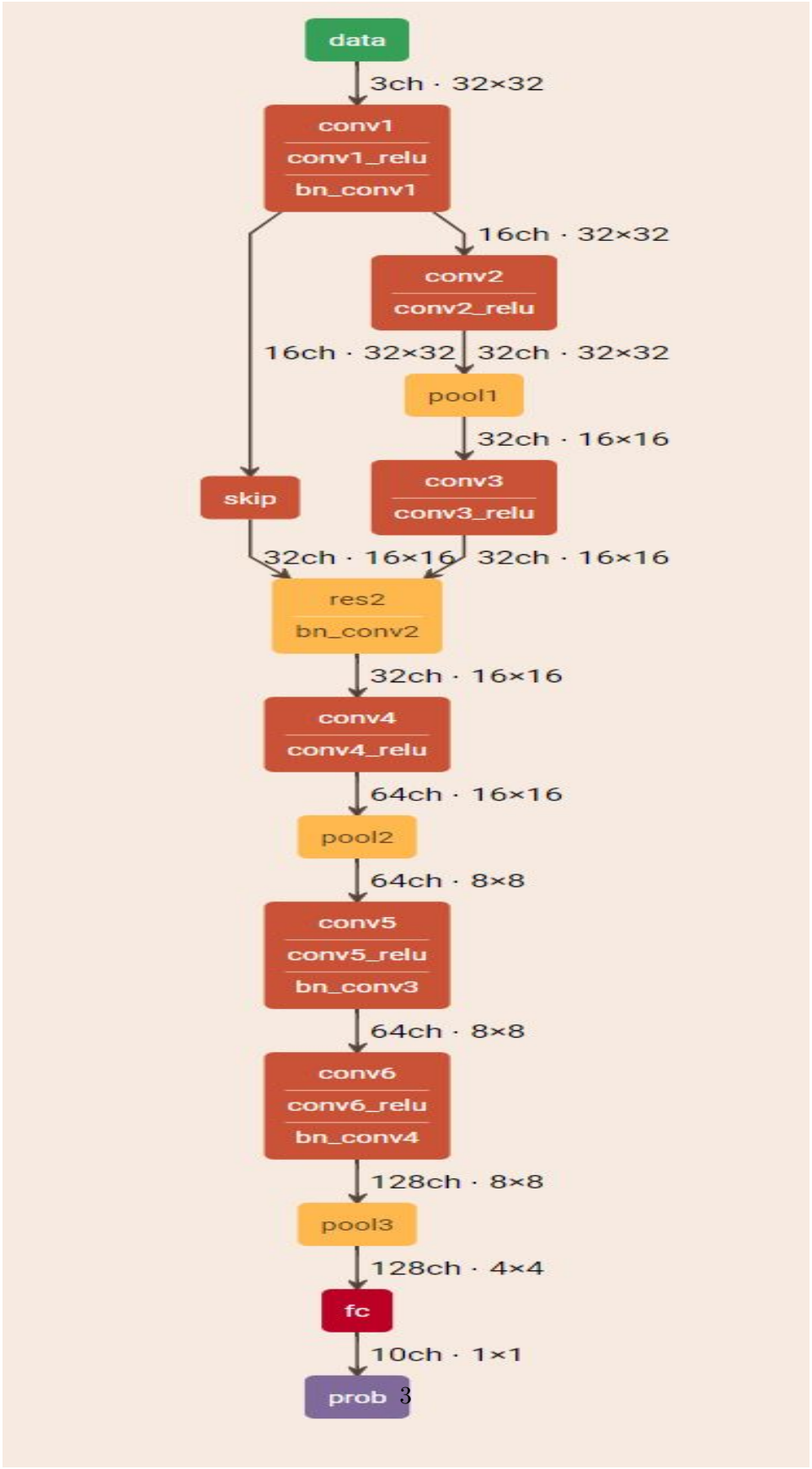

Figure 1: CNN Structure with features of ResNet. For each number it is read as the output layers and dimensions.

3 Evaluation

3.1 General

The first evalution was to use another test set to evaluate the trained model and comparing its classification output with the true label. The test set shown a final testint accuracy of 72.5% with 10000 images.

For a visual representation, group of 7 were sampled from the test set and classified with the model. The images was visually shown with a classification result. Many of them matched what human would classify them as.

Final validation was to visually present a graph of the model's loss when training. The decreasing of the loss furthur shown the model is viable in classification task of CIFAR10 images with some inaccuracies.

By sampling random images and classfiying using the trained model in comparison to their true label, there are some results shown with model classifiying result on the top and true or false when compared to the label (Figure2). The output shows that there are still space for improvement many images were not recognized. This could also be the model output produced two very probabilities and the second highest probability is the true label. Overall, the results are promising as the model identified most of them correctly.

The losses of each epoch were all recorded and plot (Figure3). The model shown a promising result with the loss decreasing every epoch.

*This project will be at this url: `https://github.com/SkyZhangX/340-Final-Project`

3.2 Results

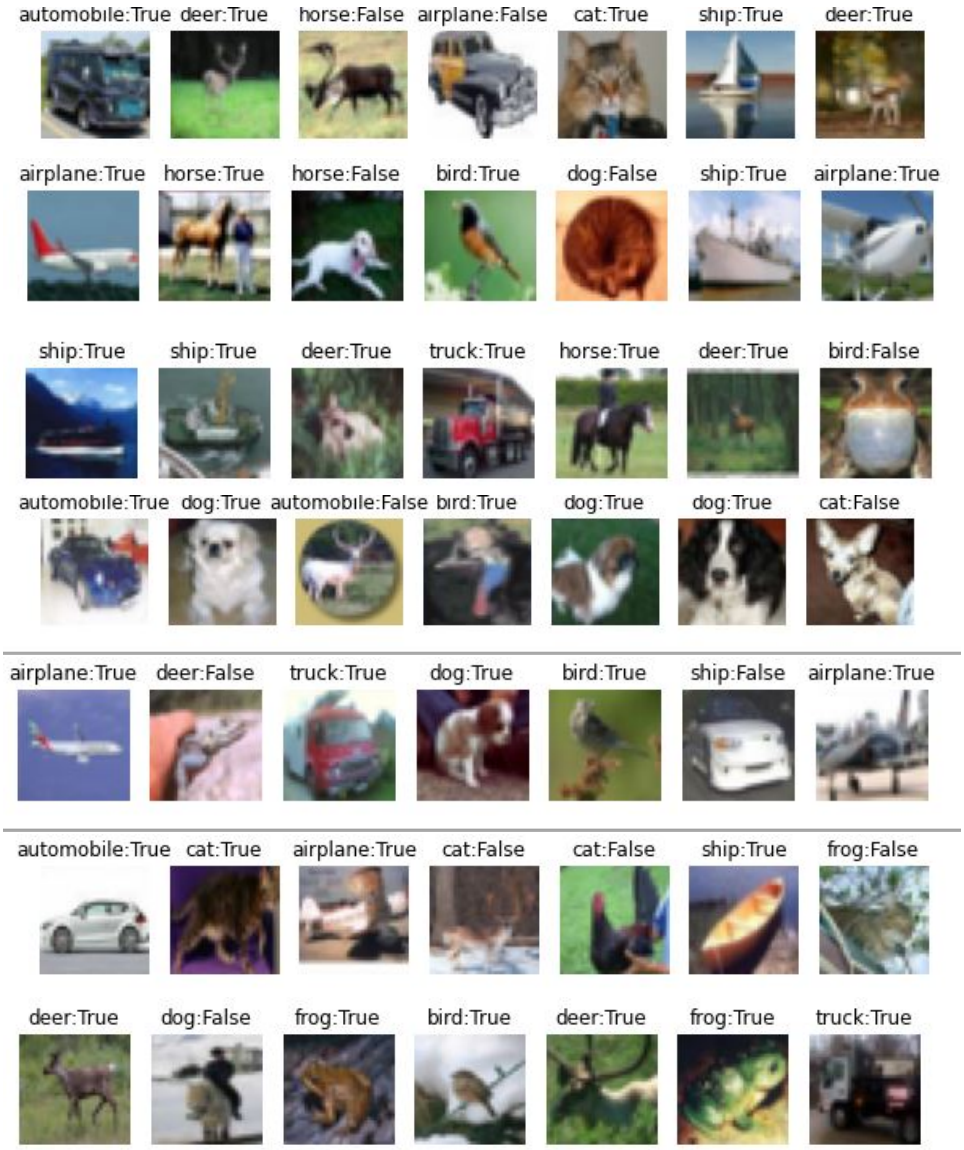

Figure 2: Samples With Classification. For a image, left top corner is the model classification, right top is True if same as label.

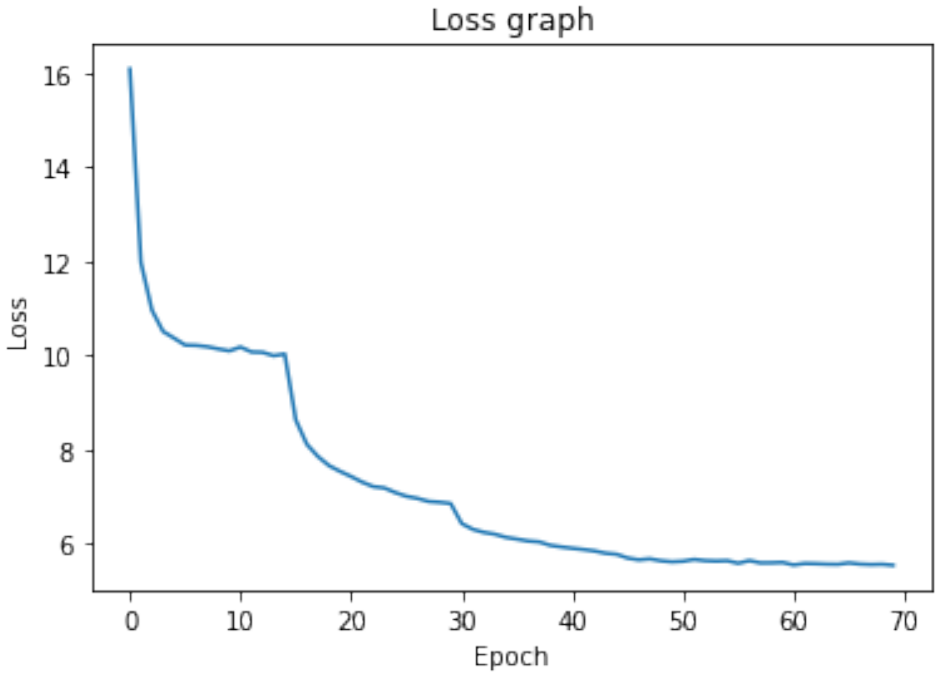

Figure 3: Loss of each epoch. The y axis are losses, the x axis are epochs.