Image Classification with CIFAR-10

Mengqi Chen (mc4398), Keran Li (kl2993), Ye Yue (yy2810), Zhen Li (zl2632)

1 Project Goals

Nowadays, deep learning plays an important role in our daily life. Image classification is a fundamental and progressive problem in the deep learning field. People try to teach computers to recognize an increasing number of types of pictures by building sophisticated convolution networks.

However, there are some difficulties in building efficient and reasonable networks as follows:
1. Too many combinations of hyper parameters -- Since we have lots of options to change the parameter, the combination of hyper parameters is infinite. Therefore, there is no straightforward method to check if we are on right way to improve the model.
2. Robustness of CNN -- The dropout function is dropping connection between neurons with a specific percentage randomly, so the result might fluctuate. If we do not set seed, the result might be different for different trials.
3. Time issue -- Training one convolution neural network is time consuming. The more sophisticated network is, the more time we will consume. Thus, efficiency is also a difficulty for training a convolutional neural network.

Our project is to use different numbers of convolutional layers, preprocessing methods and pooling methods on the CIFAR-10 dataset and compare their prediction results. By understanding different layers, we could build efficient networks when tuning the architecture for the better performance.

2 Data Preprocessing:

The CIFAR-10 dataset consists of 60 thousand color images in the size of 32x32 pixels. They are labeled in 10 classes. The labels are transportation tools and animals, including airplane, automobile, ship, truck, bird, cat, deer, dog, frog and horse.

Dataset CIFAR-10 is divided to 6 pickle files. There are 5 batches of train images and 1 batch of test images. We select first 80% images of each train batch and combine them together as train set. For the rest of images of each train batch, they are combined and consist the validation set. Corresponding to the way we split train and validation set, we use the same method to split the train and validation label so that the features and labels are consistent.

Before establishing a convolutional neural network, the data preprocessing is necessary. The main purpose of data preprocessing is to ensure the inputs have similar distribution which can fasten the convergence in training of the model. The ordinary preprocessing methods include Normalization, Zero Component Analysis and Min-Max Normalization.

2.1 Normalization

The normalization here refers to normalizing the data dimensions and making them in the same scale. To make the data zero-centered, the data needs to subtract the mean from each pixel and then divide the standard deviation of each pixel. This normalization will make the data asymptotically distributed to Standard Normal. The similar distribution of each pixels will decrease the training time of convolutional neural network model. (Karpathy 2018)

2.2 Zero Component Analysis

There are three basic properties of natural images which merit people's attention:
1. Pixels are strongly correlated to their neighbors and weakly correlated to remote pixels.
2. Pixels with the same color are highly correlated compared to pixels with different colors.
3. Pixels tend to be symmetric which means pixels at corners tend to be highly correlated.

The properties above result from the way people take pictures.

In order to avoid getting stuck in the imbalance correlation mentioned before, we need decorrelation methods. Zero Component Analysis provides a way. After using ZCA transformation, the edge information is highlighted. (Krizhevsky and Hinton 2009) Therefore, ZCA is under consideration when preprocessing data.

However, when calculating the whitening matrix, we would meet dimensional problems. We need the covariance matrix of our features. Here we collected 32x32x9000 features, which means the ZCA is hard to be handled in our case due to memory limits. (Karpathy 2018)

2.3 Min-Max Normalization

To make the normalized data between 0 and 1, the Min-Max Normalization can be used. Each pixel can be normalized by firstly minus the minimum of all the pixels and then divide the difference between the maximum and minimum of all the pixels. For image data, both [0, 1] and [0, 255] could be good for convolution. However, since the inputs after convolutional layer will be passed to the Rectified Linear Unit function, if the inputs are larger than 1, the outputs of convolution layer will be almost all 1 and the huge outputs may cause exploding gradient and terrible learning path. To make the learning steps more efficient, the data should be normalized in the range of 0 and 1.

3 Architecture Exploration and Results Display

3.1 Max Pooling

The goal of our project is image classification, so instead of collecting more features of backgrounds, we hope to focus on the figure of an image. Because background pixels are always smooth, average-pooling performs well in background feature collection. But in this case, one choice is max-pooling in our project which can keep the most expressive features. (Możejko 2017)

On one side, the Max Pooling by 2 can quickly reduce the number of parameters, thus reduce computational cost. Also, it increases the expressivity. A small change in a pixel will not change the results. It encodes a degree of variance which makes the model more robust, and avoid of overfitting.

On the other side, the size of the hidden layers is reduced too quickly, which means we cannot add too much pooling layers. But each pooling layers help us to view the image at different angle. Then, the disjoint pooling regions will limit generalization.

3.2 Fractional Max Pooling

Another choice is Fractional Max Pooling. Since NinNout (1,2), the decay rate of the hidden layers size reduces slower than applying Max Pooling and the randomness is added in the step of choosing pooling region, which makes the CNN model more elastic.

Nonetheless, there is a trade-off between the use of randomness in fractional max pooling and the use of dropout layers. We need to pay attention to the amount of dropout used. (Graham 2014)

3.3 Fully Connected

What we also apply is Fully Connected layers. Using a fully connected layer helps us to integrate all the feature from the last layer into a value, which makes the method more robust and expressive. For example, the location does not play an important role any more, and we only concentrate on the figure of this image. By the way, if our goal is image segmentation, it is unreasonable to use fully connected layer.

But if the number of fully connected layers is only one, we might cannot solve nonlinear problems. So, there is a need of more than one fully connected layer. As the number of fully connected layers increase, the model is more complicated and it performs better in nonlinear problems. We need to pay attention to overfitting problems, so there is a need for dropout and the number of fully connected layers cannot be too large. (jamesmf 2018)

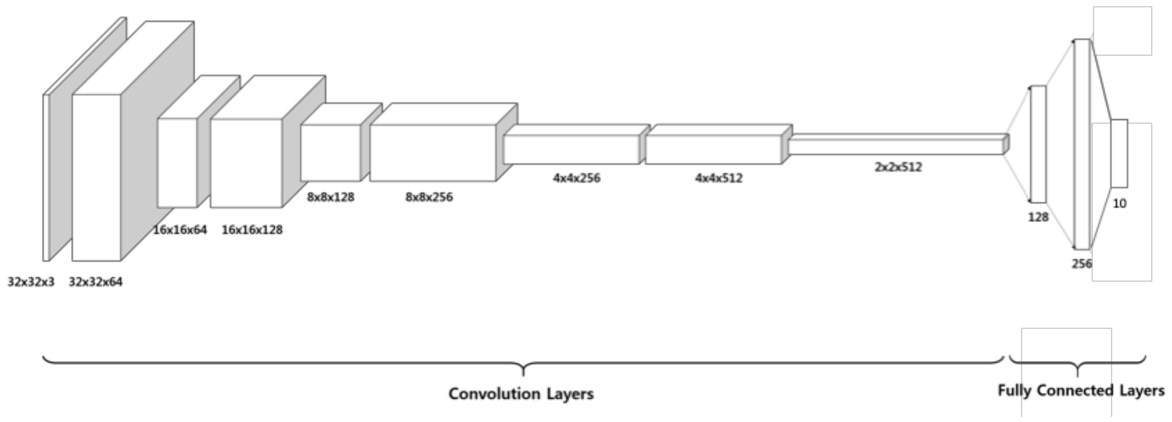
3.4 Architecture Exploration

At first, we built the 2 layers without pooling model as our baseline model. Based on the baseline model, we tuned the model by adding Max Pooling, Fractional Max Pooling and Fully Connected layers. Notice the ReLU activation function, dropout and Batch Normalization is adopted according to the demand.

The entire model contains 12 layers over all. The whole model architecture is listed as follows:

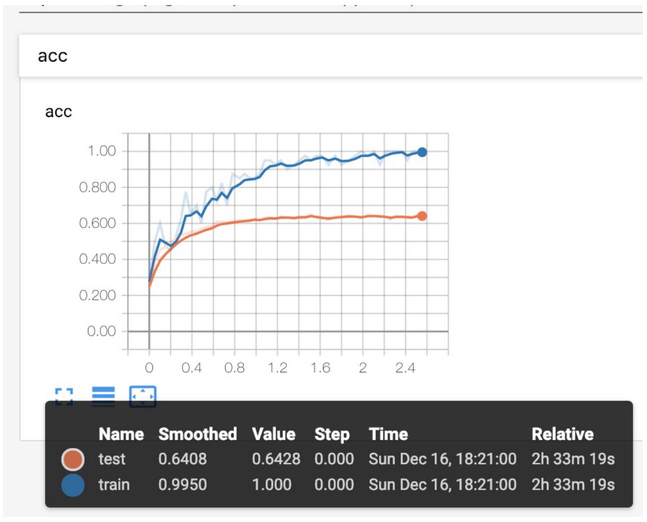
Layer 1: Convolution with 64 different filters in size of 3x3
Layer 2: Fractional Max Pooling with pooling ratio 1.44 and 1.73
Applying ReLU activation function and Batch Normalization
Layer 3: Convolution with 128 different filters in size of 3x3
Layer 4: Fractional Max Pooling with pooling ratio 1.44 and 1.73
Applying ReLU activation function and Batch Normalization
Layer 5: Convolution with 256 different filters in size of 3x3
Layer 6: Fractional Max Pooling with pooling ratio 1.44 and 1.73
Applying ReLU activation function and Batch Normalization
Layer 7: Convolution with 512 different filters in size of 3x3
Layer 8: Fractional Max Pooling with pooling ratio 1.44 and 1.73

Applying ReLU activation function and Batch Normalization
Layer 9: Flattening the 3-Dimensional output of the last convolutional operations.
Layer 10: Fully Connected Layer with 128 units with dropout and Batch Normalization
Layer 11: Fully Connected Layer with 256 units with dropout and Batch Normalization
Layer 12: Fully Connected Layer with 10 units which equals the number of classification



3.4.1 Baseline Model

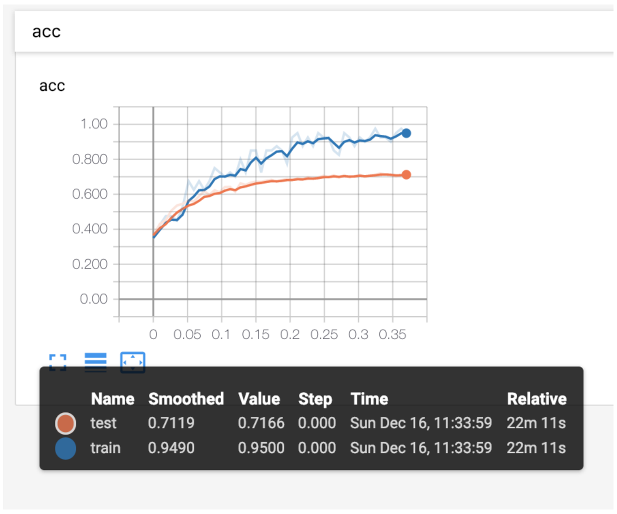
- Two Convolution Layers without Pooling



	Name	Smoothed	Value	Step	Time	Relative
●	test	0.6408	0.6428	0.000	Sun Dec 16, 18:21:00	2h 33m 19s
●	train	0.9950	1.000	0.000	Sun Dec 16, 18:21:00	2h 33m 19s

This baseline model performs not very well and takes long time. Because without pooling, the number of parameters is redundant. The performance tends to show an overfitting trend.

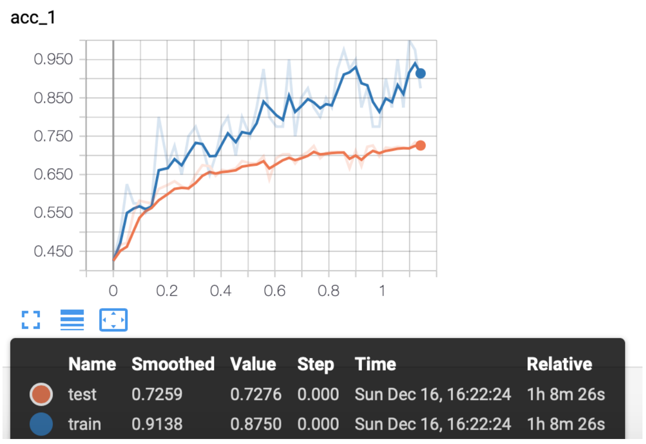
- Two Convolution Layers with Max Pooling



Name	Smoothed	Value	Step	Time	Relative
test	0.7119	0.7166	0.000	Sun Dec 16, 11:33:59	22m 11s
train	0.9490	0.9500	0.000	Sun Dec 16, 11:33:59	22m 11s

Due to the problem of redundant parameters and time issue in baseline model, as we discuss about the pooling layer above, we first try add max pooling layer after each convolution layer to maintain the most expressive feature and reduce the number of parameters to some extent. After doing this, we can see the test performance becomes better than before the train performance become worse than before. This phenomenon indicates the reduction of the overfit effects. Besides, the running time reduces significantly.

- Two Convolution Layers with Fractional Max Pooling



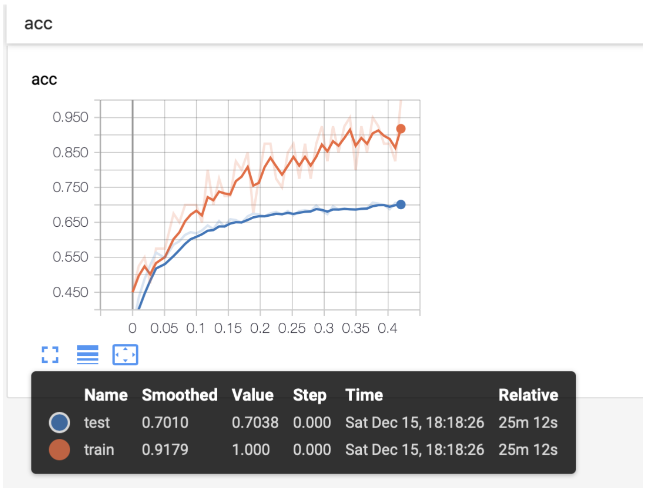
Name	Smoothed	Value	Step	Time	Relative
test	0.7259	0.7276	0.000	Sun Dec 16, 16:22:24	1h 8m 26s
train	0.9138	0.8750	0.000	Sun Dec 16, 16:22:24	1h 8m 26s

As we discuss about the advantages of fractional max pooling, we try to replace the max pooling layer with the fractional max pooling layer. According to the result of the tensorboard, there is a tiny increase in test performance. Fractional max pooling is capable of reducing the effect of overfitting as well. However, fractional max pooling consumes more running time than max pooling.

3.4.2 Three Convolution Layers

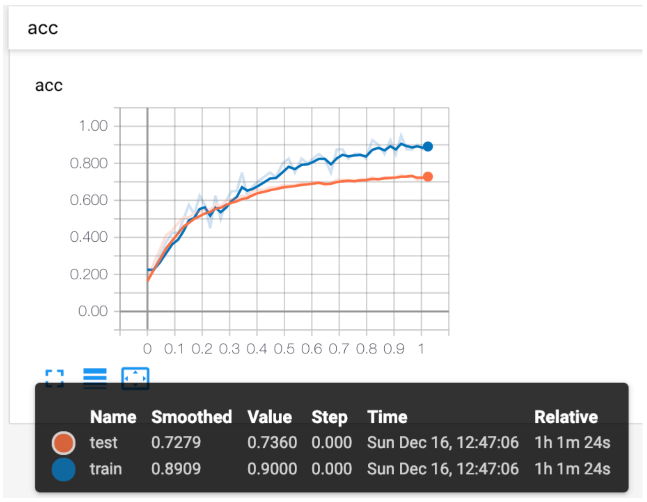
With three convolution layers, the test performance is better than that of two convolution layers under the same pooling condition, which indicates that with adding one more convolution layer, we can capture more expressive features.

- Three Convolution Layers without Pooling



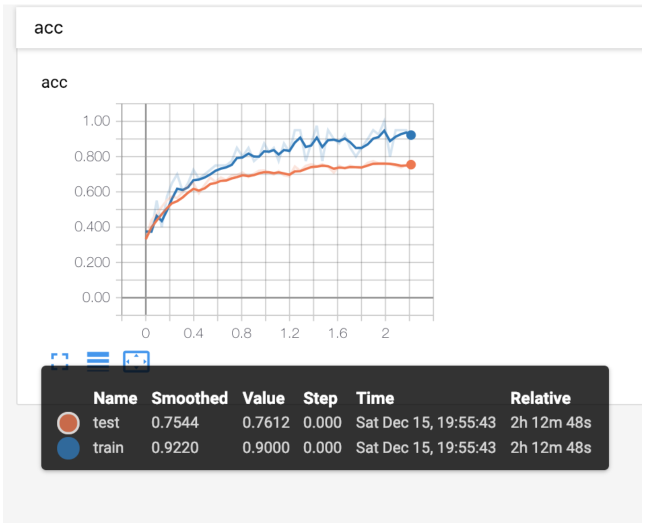
	Name	Smoothed	Value	Step	Time	Relative
●	test	0.7010	0.7038	0.000	Sat Dec 15, 18:18:26	25m 12s
●	train	0.9179	1.000	0.000	Sat Dec 15, 18:18:26	25m 12s

- Three Convolution Layers with Max Pooling



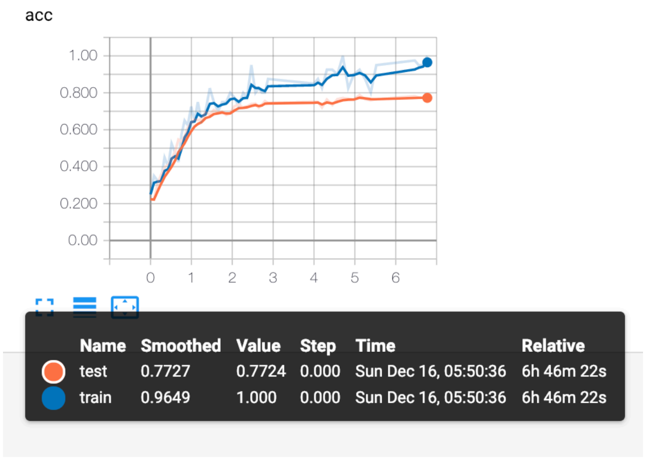
	Name	Smoothed	Value	Step	Time	Relative
●	test	0.7279	0.7360	0.000	Sun Dec 16, 12:47:06	1h 1m 24s
●	train	0.8909	0.9000	0.000	Sun Dec 16, 12:47:06	1h 1m 24s

- Three Convolution Layers with Fractional Max Pooling



	Name	Smoothed	Value	Step	Time	Relative
●	test	0.7544	0.7612	0.000	Sat Dec 15, 19:55:43	2h 12m 48s
●	train	0.9220	0.9000	0.000	Sat Dec 15, 19:55:43	2h 12m 48s

3.4.3 Four Convolution Layers with Fractional Max Pooling



	Name	Smoothed	Value	Step	Time	Relative
🟠	test	0.7727	0.7724	0.000	Sun Dec 16, 05:50:36	6h 46m 22s
🔵	train	0.9649	1.000	0.000	Sun Dec 16, 05:50:36	6h 46m 22s

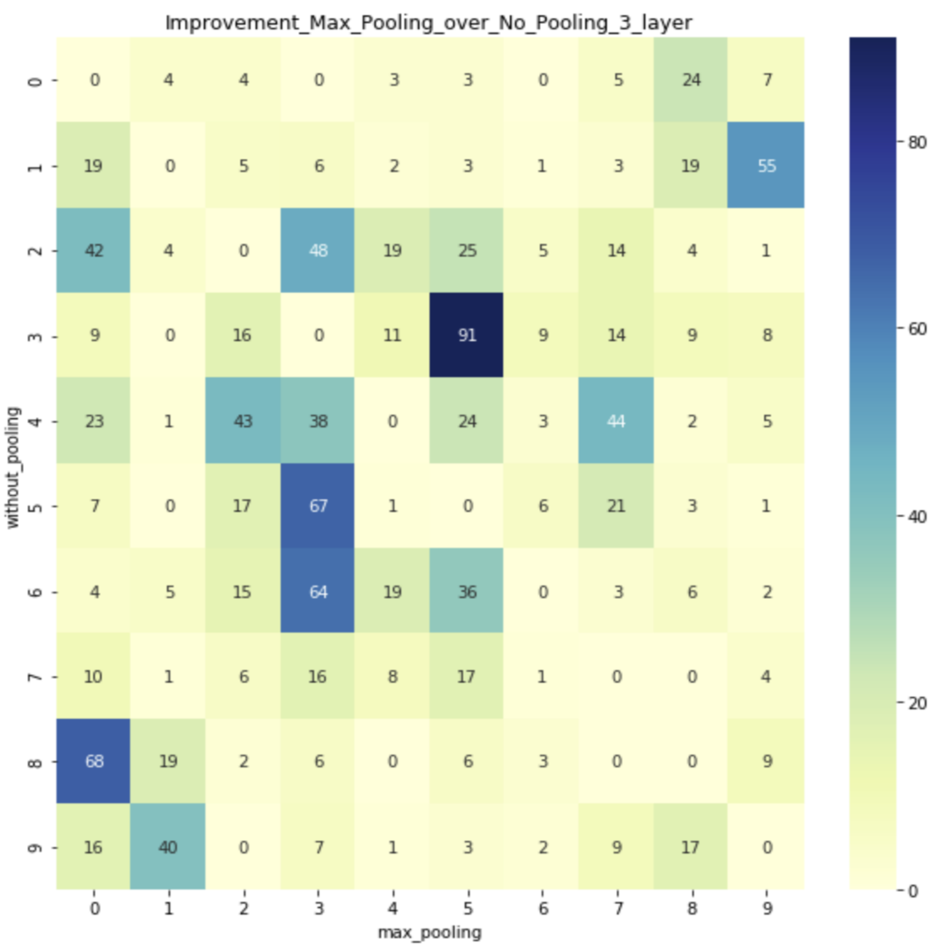
Finally, we try four convolution layers and fractional max pooling, with which we prove the improvement of the test performance. The highest accuracy of validation set is up to 0.7727 within an acceptable training time for local machine.

4 Comparison

To show the difference between models, we built confusion matrix for each 2 models by only including those misclassified by one model but correctly classified by the improved one. We apply argmax function to probability vector to get the most likely predicted label used in consist the confusion matrix.

4.1 Three Convolution Layers: Max Pooling versus No Pooling

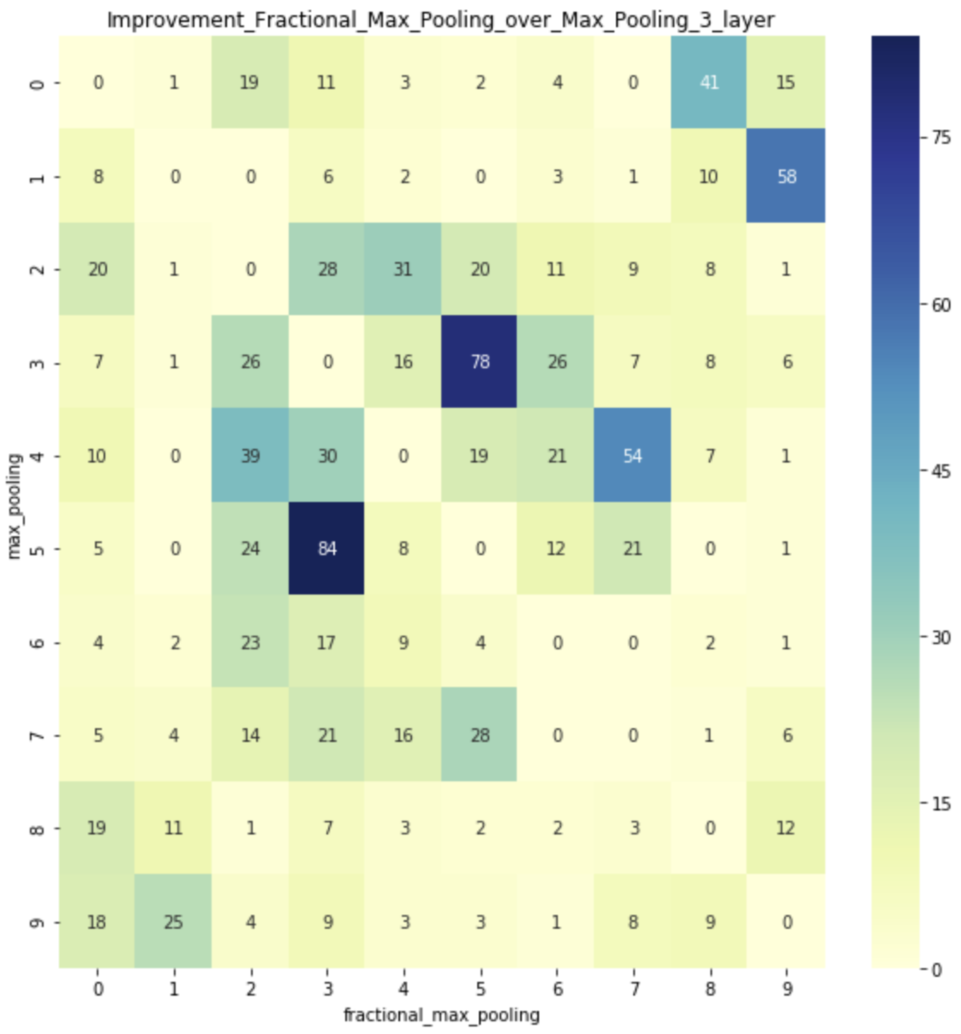
Accuracy of No Pooling: 0.6958
Accuracy of Max Pooling: 0.7288



The heatmap above shows the images misclassified by the no pooling model can be correctly classified by adding the max pooling. And we can see that the model can significantly increase the accuracy of classifying image with true label 0 (airplane), 3 (cat), 5 (dog), which are always correspondingly misclassified as 8 (ship), 5 (dog) and 6 (frog), 3 (cat).

4.2 Three Convolution Layers: Fractional Max Pooling versus Max Pooling

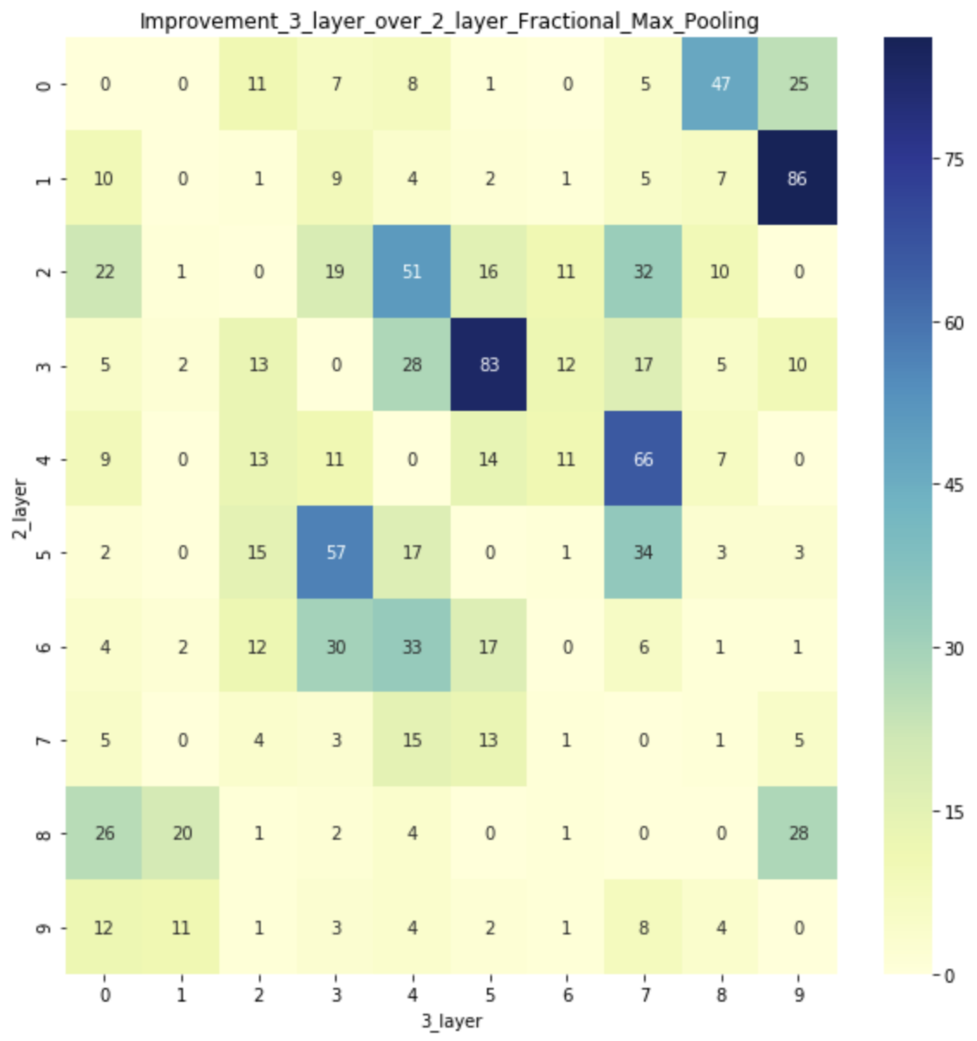
Accuracy of Max Pooling: 0.7288
Accuracy of Fractional Max Pooling: 0.7536



The heatmap above shows the images misclassified by the max pooling model can be correctly classified by using the fractional max pooling. And we can see that the model can significantly increase the accuracy of classifying image with true label 2 (bird), 3 (cat), 5 (dog), which are always correspondingly misclassified as others, 5 (dog) and 5 (dog), 3 (cat). This means fractional max pooling can tell the minor difference between cat and dog.

4.3 Fractional Max Pooling: 3 Convolution Layers versus 2 Convolution Layers

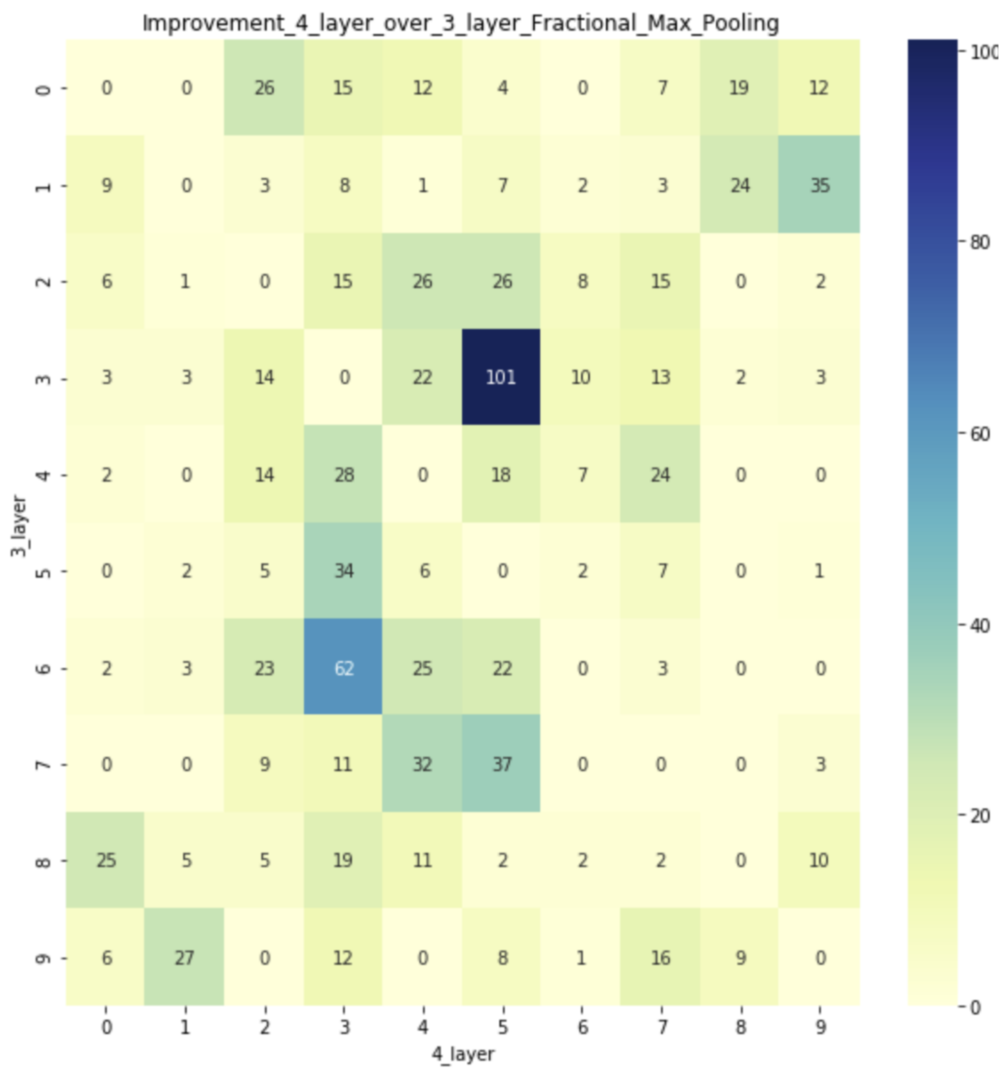
Accuracy of 2 layers: 0.7178
Accuracy of 3 layers: 0.7536



The heatmap above shows the images misclassified by the 2 layers with fractional max pooling model can be correctly classified by using the 3 layers with fractional max pooling pooling. And we can see that the model can significantly increase the accuracy of classifying image with true label 3 (cat), 4 (deer), 5 (dog), 7 (horse), 9 (truck), which are always correspondingly misclassified as others 5 (dog), 2 (automobile), 3 (cat), 4 (deer), 1 (airplane). So, adding one more layer can increase accuracy in general.

4.4 Fractional Max Pooling: 4 Convolution Layers versus 3 Convolution Layers

Accuracy of 3 layers: 0.7536
Accuracy of 4 layers: 0.7758



The heatmap above shows the images misclassified by the 3 layers with fractional max pooling model can be correctly classified by using the 4 layers with fractional max pooling pooling. And we can see that the model can significantly increase the accuracy of classifying image with true label 3(cat), 5(dog), which are always correspondingly misclassified as others, 6 (frog), 3 (cat).

5 Conclusions

After the whole data preprocessing and model tuning, we found some significant conclusions as follows:

- The best preprocessing method with affordable time consuming is Min-Max Normalization.
- The best pooling method is fractional max pooling. Compared to Max Pooling method, Fractional Max Pooling can tell minor differences between cat and dog image.
- The deeper convolutional neural network model, the higher accuracy. It can increase all categories' classification accuracy in general. The pooling methods can reduce the over fitting issues of the model.

6 Reproducibility and Improvement

All the code can be found in the GitHub *https://github.com/JannieChen/AML_Final_Project*. When reproducing the code, run the notebook according to the readme file. Each notebook contains the parts of downloading the dataset, data preprocessing, tuning the model and train and test results.

What we can do as the improvement include exploring the Fractional Max Pooling method with better pooling ratio, adding more layers to the existing model and using inception model to get more features without complicated calculation. As for inception model, it can contain several different sizes of filters, including convolution layers and pooling layers, in one inception module. (Raj 2018) Combine and concatenate the convolutions and pooling, and then send them to next inception module. Also, an inception model with dimension reduction can make the computation time more reasonable. (Mulc 2016)

Reference

Graham, B. (2014). "Fractional max-pooling." arXiv preprint arXiv:1412.6071.

jamesmf (2018). "What do the fully connected layers do in CNNs?". Retrieved 2018/12/16, 2018, from https://stats.stackexchange.com/questions/182102/what-do-the-fully-connected-layers-do-in-cnns.

Karpathy, A. (2018). "CS231n Convolutional Neural Networks for Visual Recognition." 2018, from http://cs231n.github.io/neural-networks-2/ - datapre.

Krizhevsky, A. and G. Hinton (2009). Learning multiple layers of features from tiny images, Citeseer.

Możejko, M. (2017). "Max-pooling VS Sum-pooling." Retrieved 2018/12/16, 2018, from https://stackoverflow.com/questions/37434426/max-pooling-vs-sum-pooling.

Mulc, T. (2016). "Inception modules: explained and implemented." Retrieved 2018/12/16, 2018, from https://hacktilldawn.com/2016/09/25/inception-modules-explained-and-implemented/.

Raj, B. (2018). "A Simple Guide to the Versions of the Inception Network." Retrieved 2018/12/16, 2018, from https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202.