# Advanced Machine Learning (GR5242) Final Project Topic 1: Efficient Architectures for CIFAR-10 Classifications

Yang Chen yc3335
Xinjian Hu xh2339
Ziwei Guo zg2278
Jianfeng Zhang jz2857

December 17, 2018

**Abstract**

In this project, we utilized the performance of different convolutional network architectures on the CIFAR-10 data. Here we pick four architectures that have done well in image classification. The four kinds architecture we researched on are: (1) VGG networks, (2) Network In Network, (3) Wide Residual Network, (4) Fractional Max Pooling.

In total we tried 15 varying models and got testing accuracy ranging from 50.86% up to 95.29%. Our main findings include:

- Train and test loss for the baseline logistic regression is not robust, and even unstable overtime, the basic logistic function cannot handle with the CIFAR-10 dataset perfectly.
- The test accuracy number for deeper network climbs up more slowly than that of shallow network at the very first half iterations, but raise faster than before later. Shallow network can easily reach the bottleneck for very long time, while deep network can break through the bottleneck and achieve higher precision.
- Network in network is easy to understand and could return an impressive prediction accuracy for the CIFAR-10 dataset. The dropout is necessary to avoid overfitting.
- Larger width for convolution layers especially for image processing will usually perform better since a wider layer can capture more information.
- ResNet especially one that is deep will always perform better than same player plain network.
- Data augmentation that doesn't make dramatic change to a dataset will greatly help in in the performance of deep neural network.

## Contents

## 1 Baeline Models

### 1.1 Simple Logistic Model

Here we use a basic supervised learning, logistic regression to perform the model. In this baseline model we reference the code from the lecture, using logistic regression, one-layer logistics regression, one layer with normalization layer, and etc. Below is the structure of the logistic model.

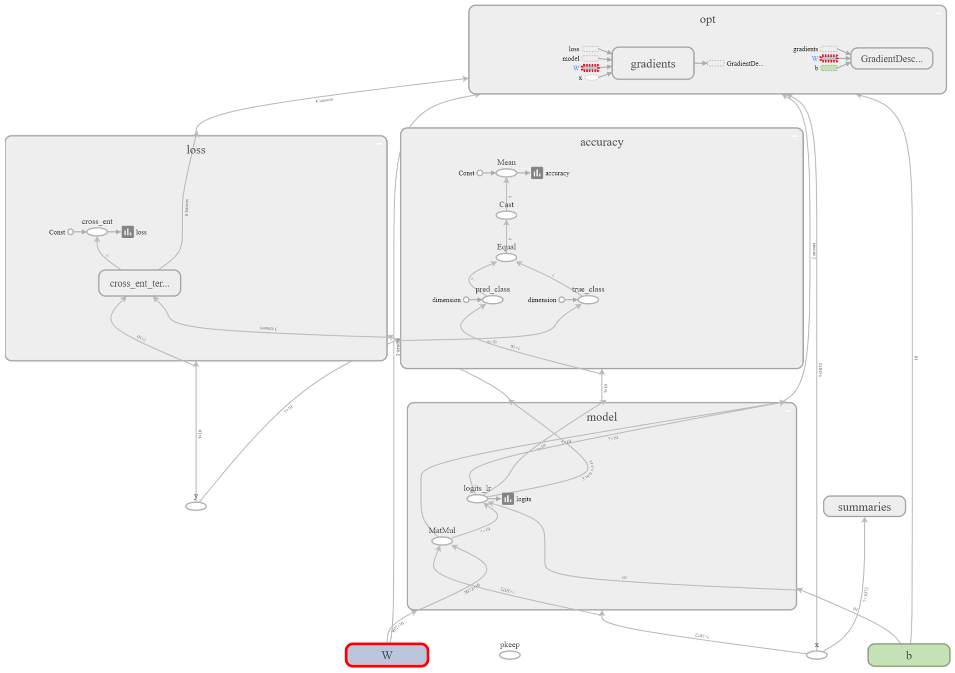
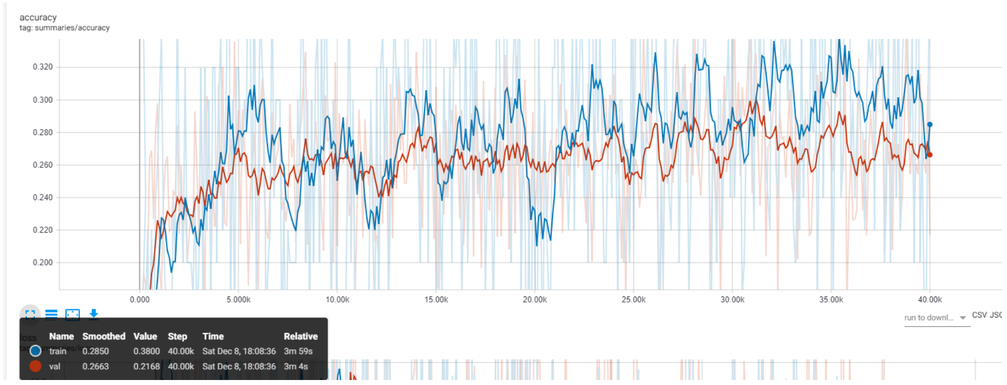


Figure 1: Structure of the Logistic Model

Then we model the CNN model, run the test, and get the accuracy for 40,000 steps iterations. We get the plot below with blue line standing for the train accuracy, with orange line standing for the test accuracy.

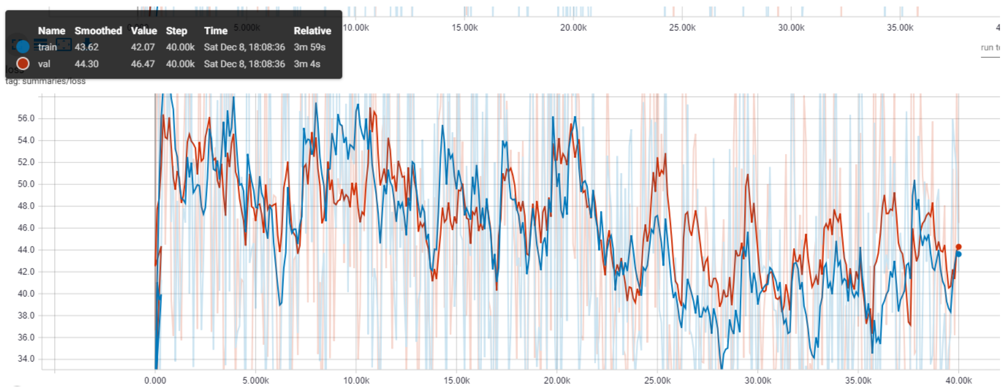


| | Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|---|
| 🔵 | train | 0.2850 | 0.3800 | 40.00k | Sat Dec 8, 18:08:36 | 3m 59s |
| 🔴 | val | 0.2663 | 0.2168 | 40.00k | Sat Dec 8, 18:08:36 | 3m 4s |

Figure 2: Accuracy of the CNN Model

We can notice that the training accuracy and the test accuracy lie within 28% to 32%, with smoothing 0.8. We also notice that from the transparent line the plot will be far more unstable, especially at the 20,000 steps. This baseline model might only work better than the random guess model (weak learner).

Below is another tensorboard graph for the cross-entropy loss function. The cross-entropy has the function $-(y\log(p)+(1-y)\log(1-p))$ and we sum up and get the function $-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$.



| | Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|---|
| ● | train | 43.62 | 42.07 | 40.00k | Sat Dec 8, 18:08:36 | 3m 59s |
| ● | val | 44.30 | 46.47 | 40.00k | Sat Dec 8, 18:08:36 | 3m 4s |

Figure 3: Cross-Entropy Loss Function

The train and test losses look not robust, the losses violate over the time and their values are extremely large.

Overall the basic logistic function cannot handle out the classification problems. And we further need to do in-depth research on more architectures of CNN.

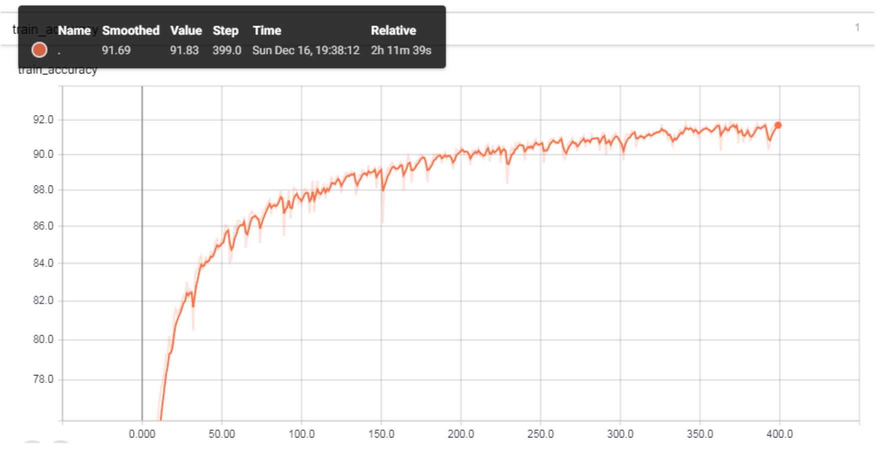
## 1.2 Simple CNN Model with Max Pooling

The network we use here has a linearly increasing number of filters per two convolutional layers. We can therefore describe the network using the shorthand form

$$(32nC2 - 32nC2 - MP2)_3 - FC - output,$$

where $32n$ indicates that the number of filters in the $n$-th convolutional layers is $32n$, and the subscript 3 indicates three pairs of $C2 - C2 - MP2$ layers. When we use dropout, we use an increasing amount of dropout the deeper we go into the network; we apply 20% dropout in the first two hidden layer, and increase linearly to 40% dropout in the final hidden layer. We use rectified linear units.

We model the above CNN model, run the test, and get the accuracy for 400 epochs. The two subfigures below are train and test accuracy, respectively. From these figures, we can find that the train accuracy is both higher and more stable than the test accuracy. The train accuracy and the test accuracy are 91.83% and 85.72%, respectively, with smoothing 0.6.



| | Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|---|
| ● | . | 91.69 | 91.83 | 399.0 | Sun Dec 16, 19:38:12 | 2h 11m 39s |

Figure 4: Train Accuracy

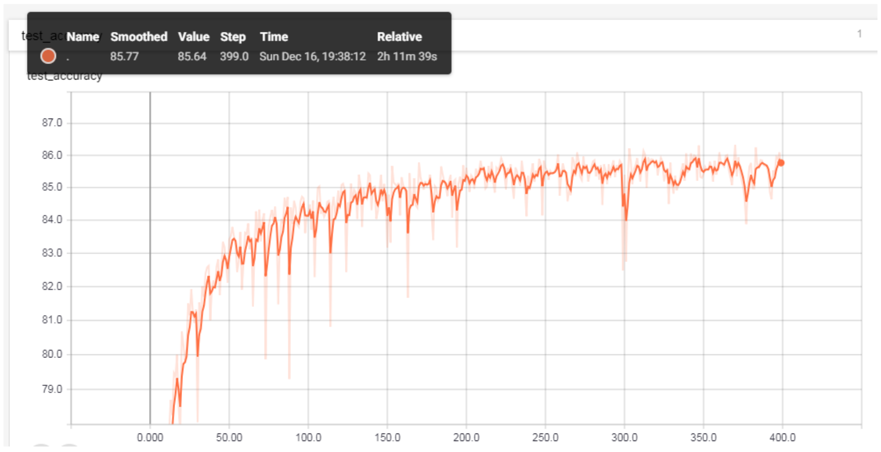


Figure 5: Test Accuracy

Then, the cross-entropy losses are shown in the two subfigures below. The train loss steps down to 0.2341 firmly with little violation, however, the test loss violates over time and ends at around 0.5131. This suggests that this model has an overfitting concern and has a poor generalization ability.

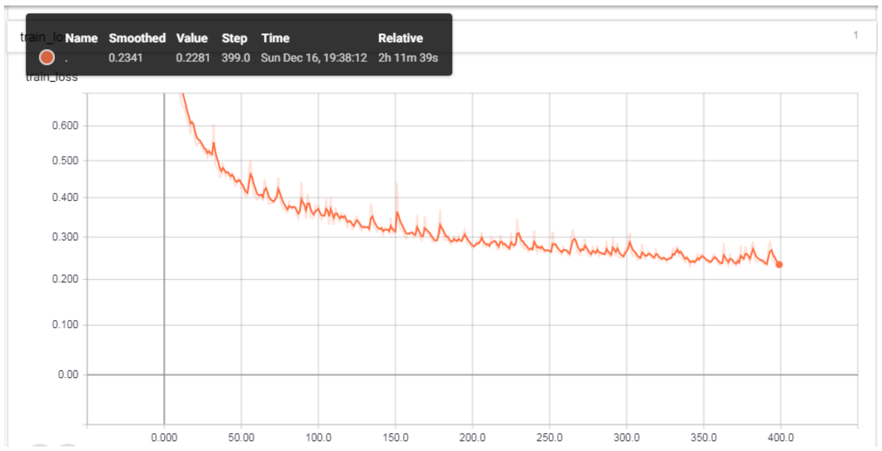


Figure 6: Train Loss

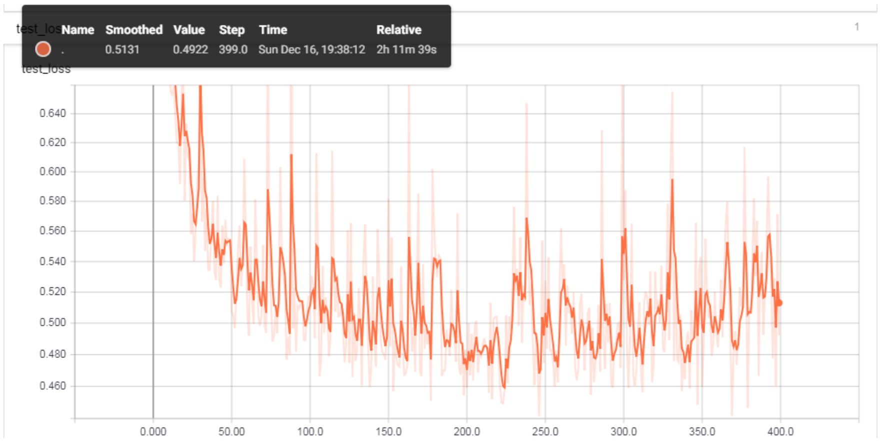


Figure 7: Test Loss

Overall the simple CNN model with max pooling cannot handle out the classification problems very well. Thus, we try to use another pooling technique to make some improvements with this simple CNN model in Section 2.4.

## 2 Architectures

### 2.1 VGG (Very Deep Neural Network)

VGG is a convolutional neural network published from researchers at Oxford's Visual Geometry Group. The model handle with the ImageNet classification challenge with 7.3% error rate. As we know, ImageNet is the most comprehensive hand-annotated visual dataset, and they hold competitions every year where researchers from all around the world compete. Here since we need only to train and test with only 10 class images, therefore, we think VGG will handle with the CIFAR-10 problems perfectly, plus VGG is remarkable for its simplicity and high accuracy.

Here we use the famous VGG-14, VGG-16 and VGG-19 models for image classification. VGG uses $3 \times 3$ convolution and $2 \times 2$ pooling throughout the network while VGG differentiates from other image classification networks by adopting very small $3 \times 3$ convolution filters and pushing the depth of the network to 14, 16, 19 weight layers. Small filters can capture more non-linear information than larger ones do. Here we compare three VGG models which show that deeper networks give better results. The detailed architectures of VGG networks are shown below.

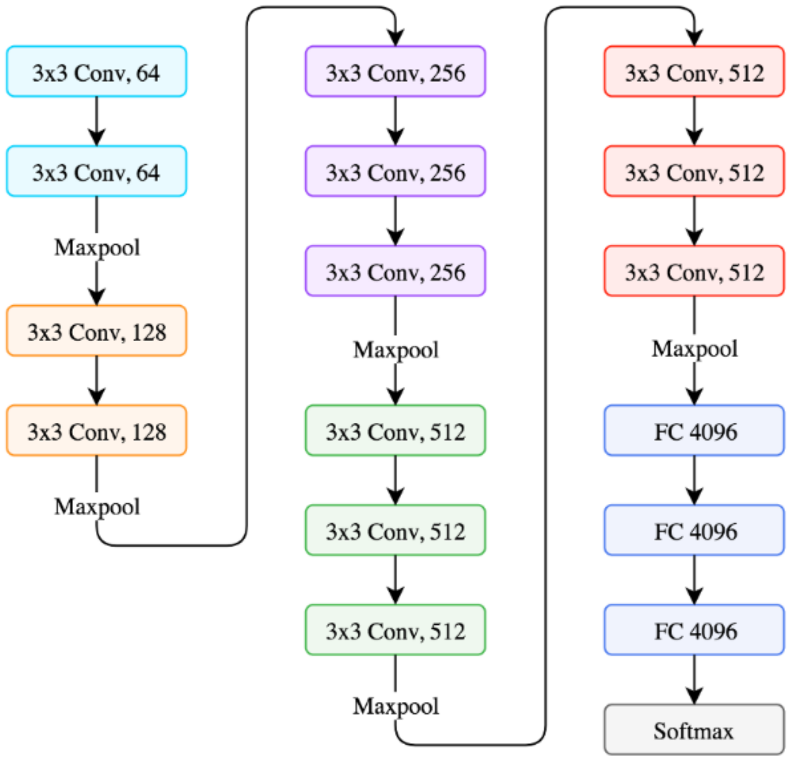


Figure 8: Architectures of VGG Networks

For the training, the input to the CIFAR-10 is a fixed-size $32 \times 32$ RGB image. For their model, they preprocess and subtract the mean RGB value. The batch size was set to 256, momentum to 0.9. The training was regularized by weight decay, with max pooling and finally dropout. VGG has been proven to perform well also on datasets with fewer classes, such as CIFAR-10. We firstly use the small VGG model from 3 to

10, then we find that the performance is poor, therefore here we diagnosed by using 14, 16 and 19. When we implemented the VGG-14 model on the CIFAR-10 data for the first time, we did not get good result. We then decided to try out part of VGG-16 network first to see how adding layers affects the performance. We decided to try tiny models first because in the reference paper of the model, the author pointed out that bad initialization can stall learning due to the instability of gradient in deep nets. So, they began with a training shallower configuration in the figure below. The left one is VGG-14, the middle one is VGG-16 while the right one is VGG-19.

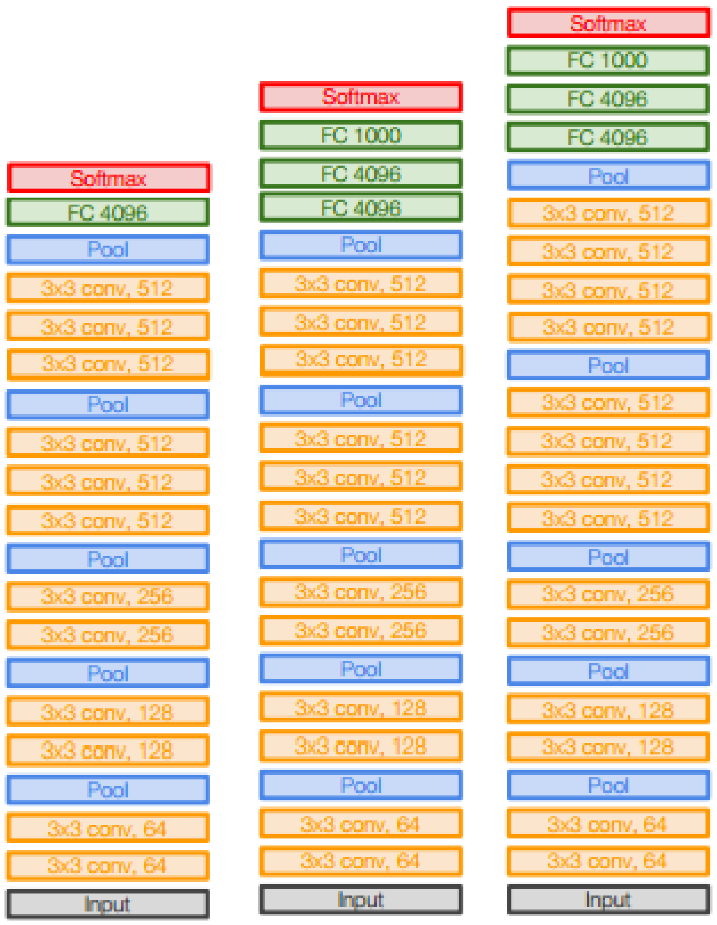
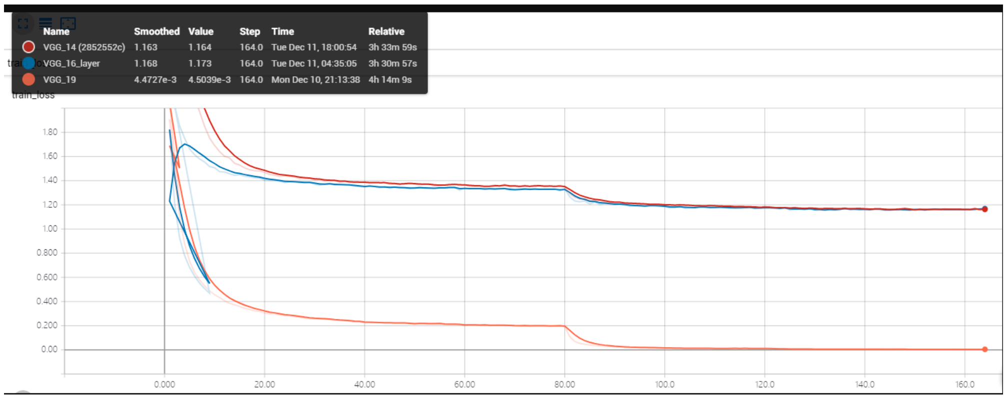


Figure 9: Shallower Configuration

Actually, the reason why we did not get desirable results with the 14 deeper networks previously was that the capability of the computation was limited, VGG networks with more than 10 layers started off very slowly and kept classifying images as the same class during the first several steps. Later on, we ran models on Colab with GPU and achieved reasonable results.

As we increased the depth, the problem of overfitting is unavoidable, since all these three models have high train accuracy. Hence, we adopted $L_2$ regularization on these models. The train accuracy of these three models almost have not much difference with 99.6% for VGG-14, 99.73% for VGG-16 and 99.97% for VGG-19.

Below is the regularization train loss.



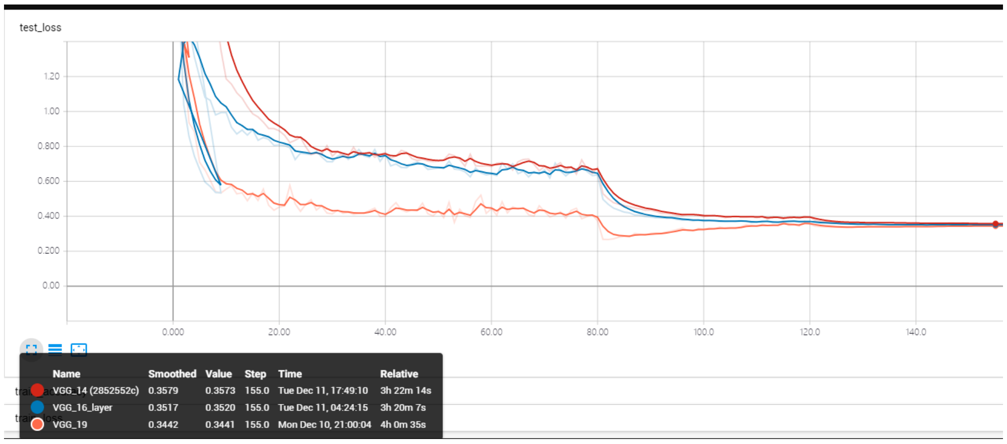
| | Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|---|
| ● | VGG_14 (2852552c) | 1.163 | 1.164 | 164.0 | Tue Dec 11, 18:00:54 | 3h 33m 59s |
| ● | VGG_16_layer | 1.168 | 1.173 | 164.0 | Tue Dec 11, 04:35:05 | 3h 30m 57s |
| ● | VGG_19 | 4.4727e-3 | 4.5039e-3 | 164.0 | Mon Dec 10, 21:13:38 | 4h 14m 9s |

Figure 10: Train Loss

Notice in the last few epochs, the train loss of the VGG-19 approaches 0 which further proves that VGG-19 works well on the CIFAR-10 than VGG-14 and VGG-16.

The most important, test accuracy tensorboard shows us that all these three models perform well and do not overfit on the CIFAR-10.

Nevertheless, unlike the train losses, the test losses of these three models converge to the same value. This might suggest that VGG-19 might have an overfitting concern.



| | Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|---|
| ● | VGG_14 (2852552c) | 0.3579 | 0.3573 | 155.0 | Tue Dec 11, 17:49:10 | 3h 22m 14s |
| ● | VGG_16_layer | 0.3517 | 0.3520 | 155.0 | Tue Dec 11, 04:24:15 | 3h 20m 7s |
| ● | VGG_19 | 0.3442 | 0.3441 | 155.0 | Mon Dec 10, 21:00:04 | 4h 0m 35s |

Figure 11: Test Loss

In summary, the VGG models perform well on the CIFAR-10, but unlike VGG perform on ConvNet project which have 96% test accuracy we perform not good as them, the reason might be, the reason of small and large width of convolution layer. Since the input of the original VGG model is $224 * 224 * 3$ while the CIFAR-10 model is $32 * 32 * 3$, even though we shrink the width of the convolution network, the wider convolution layers performed better because they have more information. Furthermore, the deeper the network, of course the more the test accuracy we have and the more the time consumed to compute, and all these VGG models reach the "bottleneck" area at the same time.

## 2.2 Network in Network

The second model used to predict classes of the CIFAR-10 dataset is Network in Network. Instead of conventional convolutional layer using linear filters followed by a nonlinear activation function, Network in Network used micro neural networks (MLP) with more complex structures.

The overall structure of NIN is a stack of mlpconv layers, on top of which lie the global average pooling and the objective cost layer. The comparison of structures of linear convolution layer and mlpconv layer are shown below. Sub-sampling layers can be added in between the mlpconv 4 layers as in CNN and MaxOut networks.

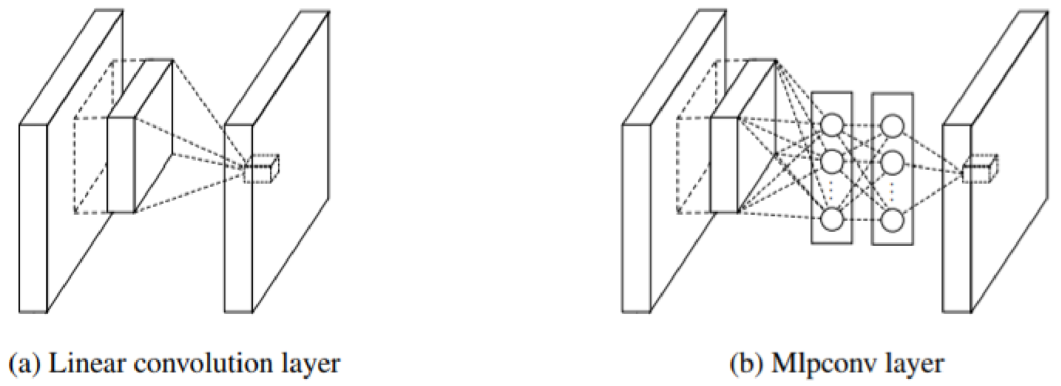


Figure 12: Comparison of Structures of Linear Convolution Layer and MLPCONV Layer

As for training the model, input data is standardized firstly. Then, the figure below shows the architecture of NIN used in this paper with three mlpconv layers. Within each mlpconv layer, there is a three-layer perceptron and pooling. We used average pooling for the first two mlpconv layers and global average pooling for the last mlpconv layer. The global average pooling here is used to replace the fully-connected layer in traditional CNN, which helps generate a feature map for each corresponding category of the classification task in the last mlpconv layer. Besides, dropout is used between each mlpconv layer. Also, a softmax logistic regression layer is added at the end.

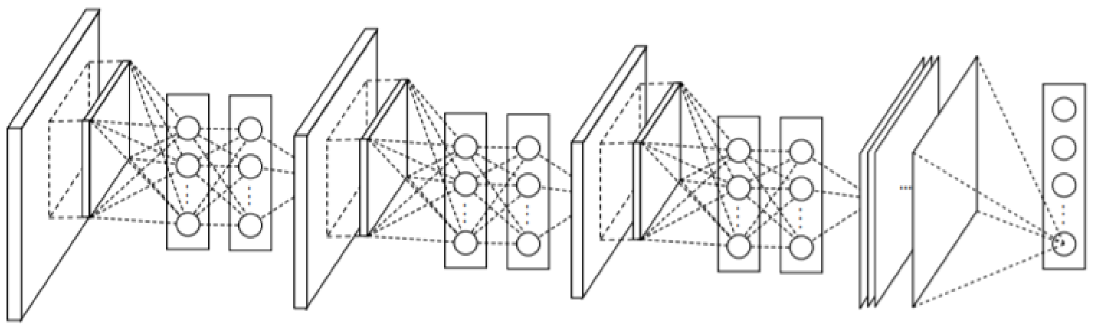


Figure 13: Architecture of NIN

What's more, data augmentation is utilized here. Since the paper reports that there is 1% decreasing of accuracy when using data augmentation.

When training the data, we firstly set batch size to be 128, epoch to be 100 and iterations to be 400. And this model returns 95.91% accuracy for training set and 89.33% accuracy for validation set. Train loss is shown by tensorboard as below. We could find that it decreases from 1.6425 to 0.2540 with the increase of epochs from 0 to 200. For test loss, it decreases from 1.6859 to 0.5335.

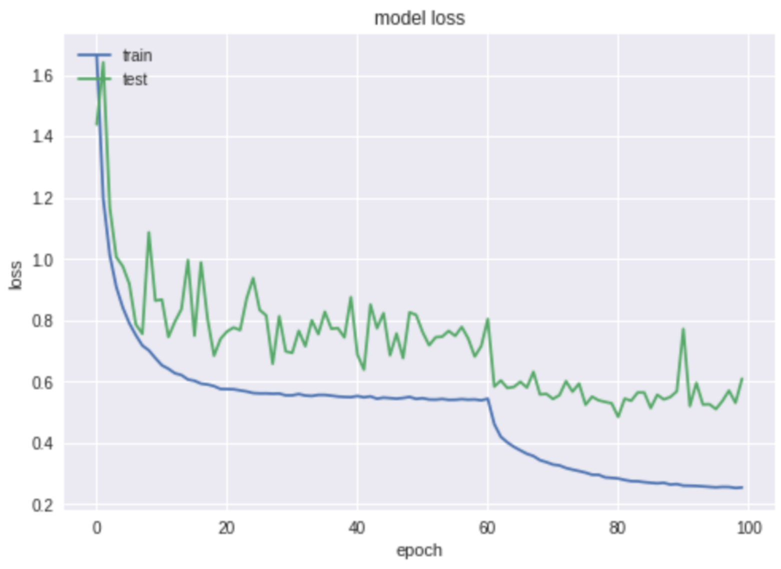


Figure 14: Train and Test Loss

However, the train accuracy shows that the accuracy for train set converged to 0.89 from 60 epochs and the test accuracy converged to 0.8933, which can tell that there doesn't exist overfitting.

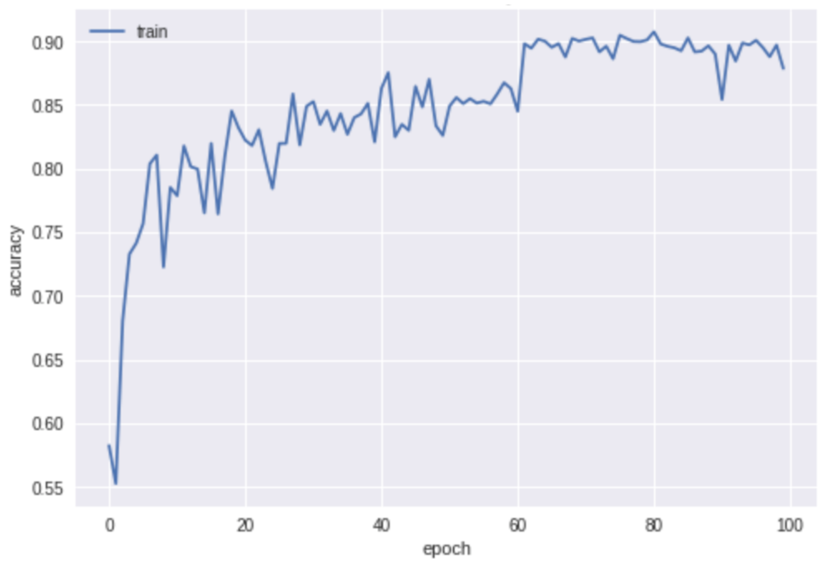


Figure 15: Train Accuracy

To conclude, the architecture of network in network is easy to understand and could return an impressive prediction accuracy for the CIFAR-10 dataset. And also, the dropout method is utilized in the model to avoid overfitting. However, compared to VGG model, Network in Network took much longer to run and returned a lower accuracy even though hyperparameters in the model were modified.

## 2.3 Wide Residual Network

This is basically a normal ResNet with more feature maps and some other tweaks. It is known that residual network could be built without losing the accuracy. As for this network, the depth of a neural network refers the number of layers and the width refers to the number of neurons per layer, or for the convolutional layers. In addition, a wide layer can learn more different features which also requires to optimize more parameters than the normal neural network. One of the biggest advantages for this model would be more vulnerable to overfitting. In conclusion, a wide ResNet is the residual network which has more feature maps in its convolutional layers.

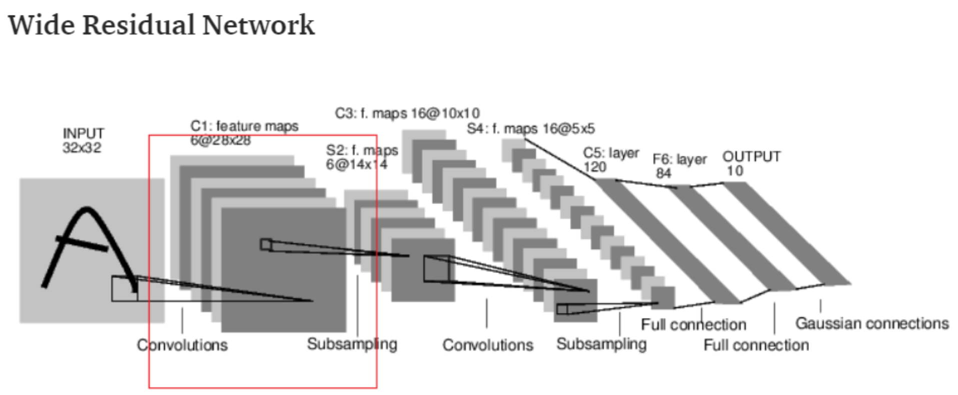
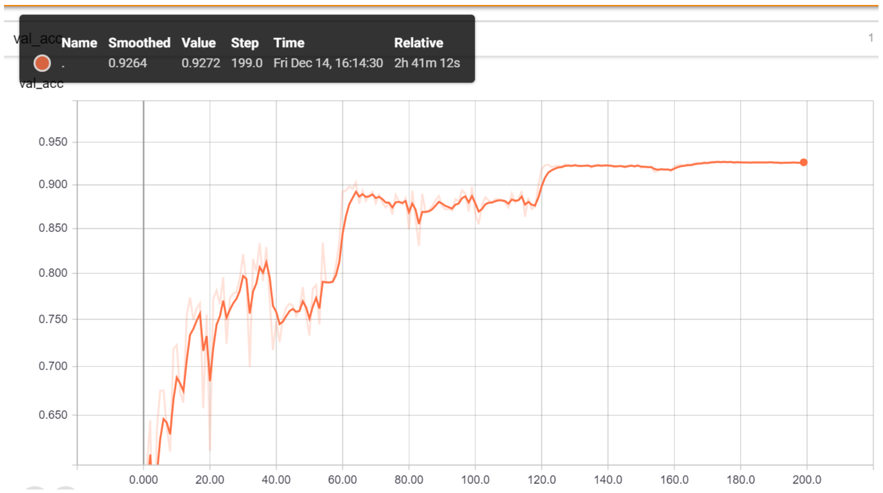


Figure 16: Architecture of Wide ResNet

From this picture, the part in the red box shows the difference that the convolutional layer has increased number of feature maps. There are 10 convolutional layers for the model. The batch size here is set as 128 and the number of epochs is 200. Other parameters like the number of iterations is about 232 and the weight of decay is set as 0.0005.

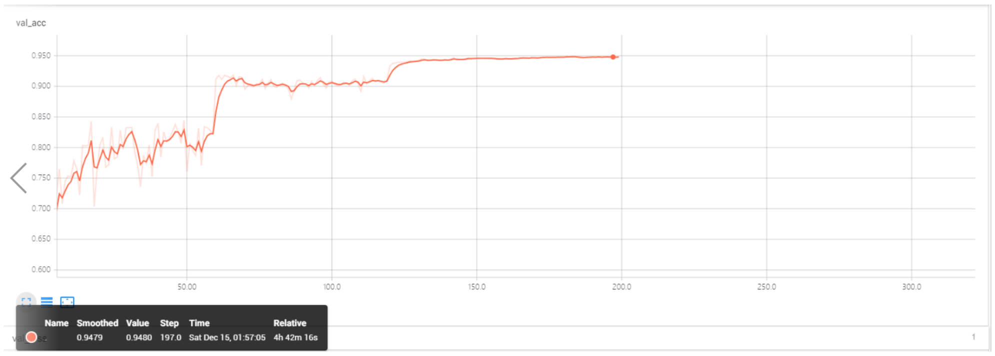
For the CIFAR-10 dataset, two different wide ResNet has been tested. Firstly, a 14 × 6 wide ResNet has been trained with a great result. The training accuracy is about 93.21% which indicates the model has greatly explain the dataset. The test accuracy is 92.72% and loss is 0.3920.



| Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|
| . | 0.9264 | 0.9272 | 199.0 | Fri Dec 14, 16:14:30 | 2h 41m 12s |

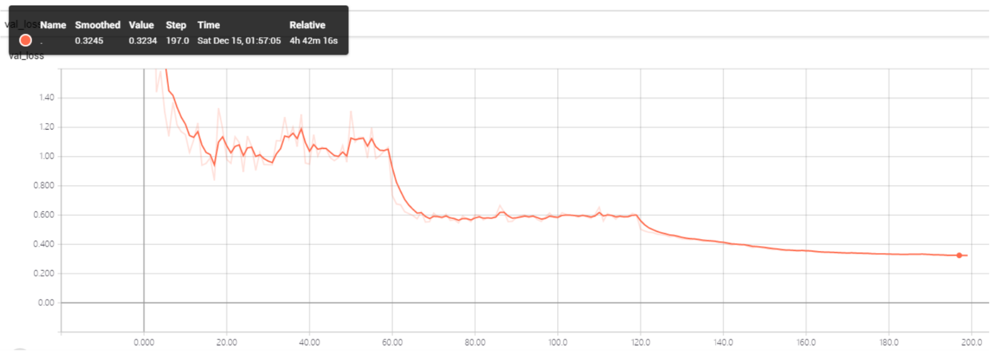
Figure 17: Test Accuracy of the First Wide ResNet

The second wide ResNet has changed the depth to 16 which outperforms than the first one. But other parameters would keep the same as the 14-depth model. To be more precise, the training accuracy becomes 99.98% which is almost about 100 percent. The test accuracy also has a little increase to 94.81% and loss is decrease to 0.3229. Since there is not much difference between test accuracy and train accuracy, there is no overfitting problem under this model.



| Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|
| . | 0.9479 | 0.9480 | 197.0 | Sat Dec 15, 01:57:05 | 4h 42m 16s |

Figure 18: Test Accuracy of the Second Wide ResNet



| Name | Smoothed | Value | Step | Time | Relative |
|---|---|---|---|---|---|
| . | 0.3245 | 0.3234 | 197.0 | Sat Dec 15, 01:57:05 | 4h 42m 16s |

Figure 19: Test Loss of the Second Wide ResNet

Since this network has more feature maps which cost more time to train, the final results are better than the algorithms which have been implemented above. As a result, it is optimized well to classify the target picture.

## 2.4 Fractional Max Pooling

The focus of this section is a particular form of max-pooling that we call fractional max-pooling (FMP). The idea of FMP is to reduce the spatial size of the image by a factor of $\alpha$ with $1 < \alpha < 2$.

Each convolutional filter of a CNN produces a matrix of hidden variables. The size of this matrix is often reduced using some form of pooling. Max-pooling is a procedure that takes an $N_{in} \times N_{in}$ input matrix and returns a smaller output matrix, say $N_{out} \times N_{out}$. For regular $2 \times 2$ max-pooling, $N_{in} = 2N_{out}$. In this case, $N_{in}/N_{out} \approx 2$ so the spatial size of any interesting features in the input image halve in size with each pooling layer. In contrast, if we take $N_{in}/N_{out} \approx \sqrt[n]{2}$ then the rate of decay of the spatial size of interesting features is $n$ times slower. For clarity we will now focus on the case $N_{in}/N_{out} \in (1, 2)$ as we are primarily interested in accuracy; if speed is an overbearing concern then FMP could be applied with $N_{in}/N_{out} \in (2, 3)$.

The network we use here has a linearly increasing number of filters per two convolutional layers. We can therefore describe the network using the shorthand form

$$(32nC2 - 32nC2 - FMP\sqrt{2})_3 - FC - output,$$

$32n$ indicates that the number of filters in the $n$-th convolutional layers is $32n$, and the subscript 3 indicates three pairs of $C2 - C2 - FMP\sqrt{2}$ layers. When we use dropout, we use an increasing amount of dropout the deeper we go into the network; we apply 20% dropout in the first two hidden layer, and increase linearly to 40% dropout in the final hidden layer. We use rectified linear units.

The above model has been trained for the CIFAR-10 dataset. The following two figures are the train accuracy and the test accuracy, respectively.

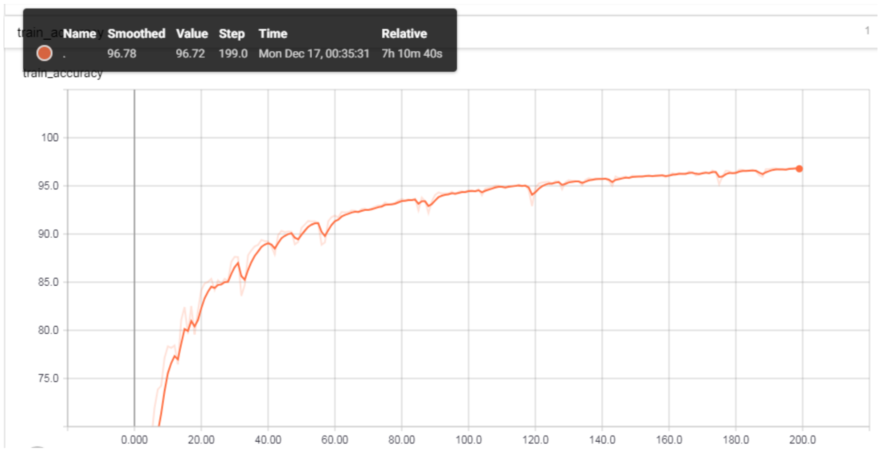


Figure 20: Train Accuracy

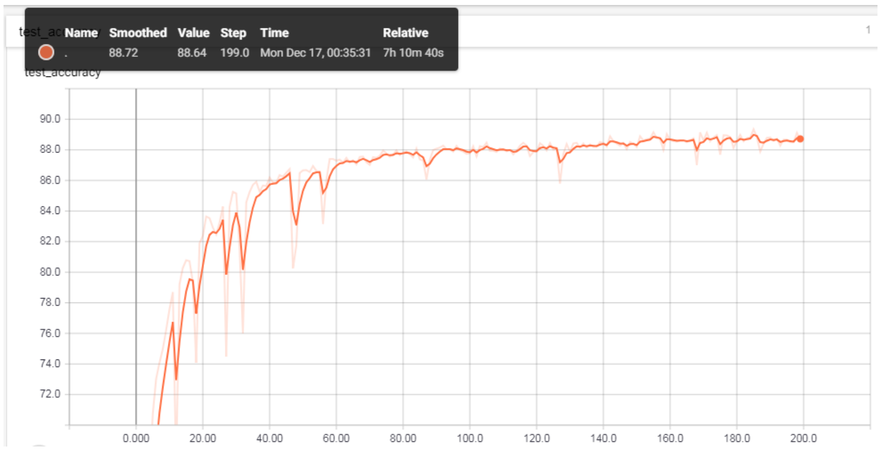


Figure 21: Test Accuracy

When comparing with the Simple CNN Model with Max Pooling in Section 1.2, we find that both the train accuracy and the test accuracy become much lower, which are 96.72% and 88.64%, respectively. In addition, the test accuracy has become much more stable than that of the previous model, which indicates that the slower rate of decay of the spatial size of interesting features introduced by Fractional Max Pooling has improved the generalization ability of this Simple CNN model to a great extent.

As shown below, both of the train loss and the test loss become smaller. The train loss is 0.0931 and the test loss is 0.4468 while that of the previous model are 0.2281 and 0.5131. This indicates that the FMP has improved the performance of the simple CNN model on both train and test sets.

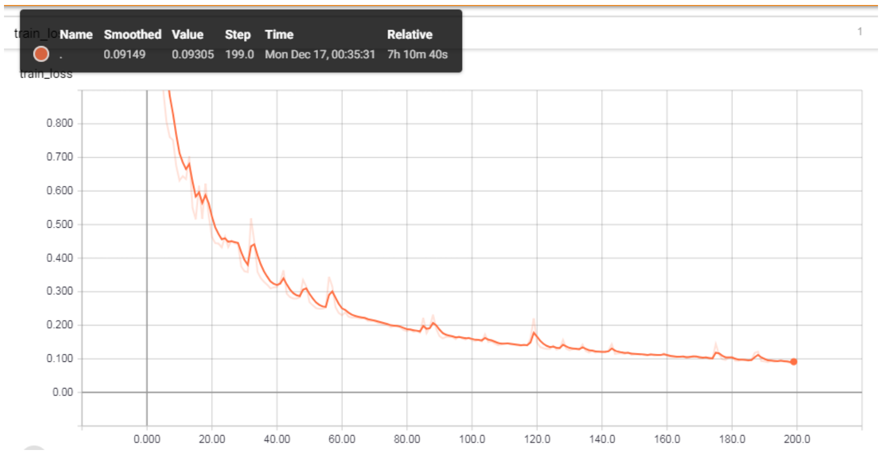


Figure 22: Train Loss

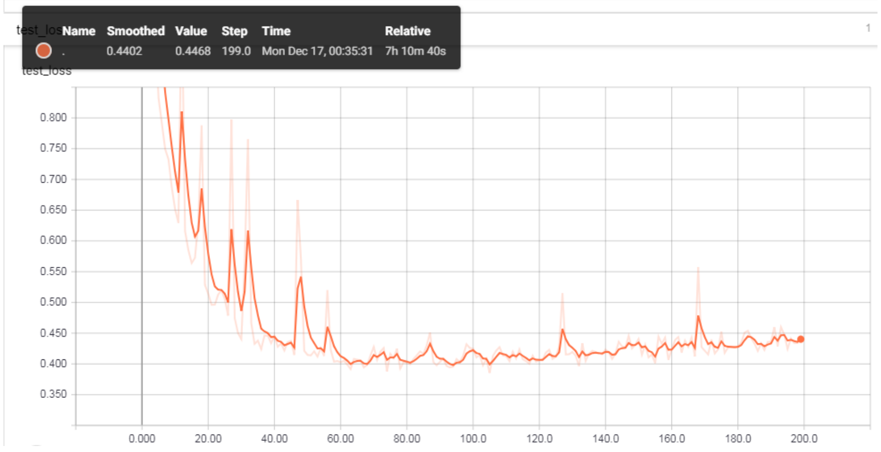


Figure 23: Test Loss

However, we can find that in both models, the train accuracies and the test accuracies are not close, which indicates that there may be an overfitting problem. Thus, we can conclude that FMP can somewhat improve the accuracy of the model but cannot deal with overfitting problems.

Finally, when compared all models mentioned before, the FMP model consumes much more time to be trained, which took 7 hrs 10 mins. This is because the fractional max pooling layers slow the rate of decay of the spatial size of the images.

In summary, the FMP model makes a big improvement compared with the Simple CNN Model in Section 1.2, but it may have an overfitting concern and it takes a much longer time to be trained.

## 3 Conclusion and Future Goals

To sum up, we have the following table to summary our four network architectures.

| Network | Params | Batch Size | Epoch | Training Time | Test Accuracy |
|---|---|---|---|---|---|
| Vgg16-Network | 138M | 128 | 200 | 3 hrs 40 mins | 93.22% |
| Vgg19-Network | 143M | 128 | 200 | 4 hrs 10 mins | 93.42% |
| Network-in-Network $16 \times 6$ | 0.97M | 128 | 200 | 3 hrs 48 mins | 89.33% |
| Wide-ResNet $14 \times 6$ | 2.7M | 128 | 200 | 3 hrs 42 mins | 92.64% |
| Wide-ResNet $16 \times 6$ | 6.6M | 128 | 200 | 4 hrs 42 mins | 94.79% |
| Fractional-Max-Pooling-Network | 1.5M | 128 | 200 | 7 hrs 10 mins | 88.72% |

Table 1: Summary of the Four Network Architectures

VGG-19 have greatest parameters and might cost a lot to memory because of the depth of its convolution layers, however with the depth of the layers, overfit might be a concern even if adopted the $L_2$ penalty; with the depth increase the test error will increase as well, though the accuracy increases not much, round 0.0046 for additional 2 or 3 layers. We might try dropout method or other regularization method to avoid overfitting in the future.

Network-in-Network have fewer parameters, which might not have a large test accuracy as VGG did, after changing the batch size and epochs size the test accuracy increase, however compared to VGG and Wide ResNet it still has low accuracy. Drop out was used here hence unlike VGG there is no need for concerning overfit. Since using google Colab will be disconnected over 5 hours therefore, it is unstable to reproduce the NIN with 256 batch size, hence we might consider to run the NIN with more powerful platform, GCP or AWS for etc.

Wide residual Network may not have much parameters as other method like VGG. However, it does perform very well. Especially for the $16 \times 6$ wide ResNet, it can perfectly explain the training dataset while have test accuracy as 94.80%. In addition, this ResNet is not hard to have a nice understanding of it which is basically a residual network with increased number of feature maps in convolutional layers. From the parameters that have been tested, it could be possible to grow the test accuracy with the increase of the depth and width. Because it has made a significant difference when there is not much change for the depth from 14 to 16. In conclusion, there is further information and value that we could explore for the wide residual network.

Fractional Max Pooling do not have many parameters. It can be used to improve the performance of existing CNN models; however, the cost is that it will take a much longer time to be trained. And, in the FMP in this paper, we use $32n$ filters in each $n$-th layer due to the limit of the GPU in Colab, however, we can use more filters to improve the accuracy of our model. In addition, we can try to use different fractional pooling rates in each layer to find the most appropriate parameters for the CIFAR-10 dataset.

## References

[1] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout Networks. arXiv: 1302.4389, 2013.

[2] Min Lin, Qiang Chen, Shuicheng Yan. Network In Network. arXiv:1312.4400, 2013.

[3] Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015.

[4] Sergey Zagoruyko, Nikos Komodakis. Wide Residual Networks. arxiv: 1605.07146, 2016.

[5] Benjamin Graham. Fractional Max-Pooling. arXiv:1412.6071, 2014.

[6] Erik Hvass Pedersen. TensorFlow Tutorial CIFAR-10. https://github.com/ Hvass-Labs/TensorFlow-Tutorials/blob/master/06 CIFAR-10.ipynb

[7] Kaggle. https://www.kaggle.com/c/cifar-10/discussion/40237

[8] Daniil Pakhomov. Image Classification and Segmentation with Tensorflow and TF-Slim. http://warmspringwinds.github.io/tensorflow/tf-slim/2016/10/30/image-classification-and-segmentation-using-tensorflow-and-tf-slim/

[9] Keras team. VGG19. https://github.com/keras-team/keras/blob/master/ keras/applications/vgg16.py