Advanced Machine Learning (GR5242) Final Project

December 17, 2018

Fangqi Ouyang(fo2203)

Yinan Wang(yw3035)

Yaxin Wang(yw3042)

Yanchen Chen(yc3373)

1. Introduction

In the project, we utilize and modify the convolutional neural network on CIFAR-10 dataset. This dataset is popular and accessible to the public. It has ten categories including airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There are 50000 pictures in the training set and 10000 pictures in the testing set. After researching a few convolution neural network architectures that performed well on classifying accuracy, we implement our own deep learning model on CIFAR-10 dataset based on these architectures, which are (1) Very deep convolutional network, (2) Network in network, and (3) Wide residual network. We start with classical convolution neural network as a baseline architecture and apply different designs to improve classification performance. After trying several architectures and changing parameter values, the test accuracy starting from 67.27% has been improved to 92.81%。

2. Architectures

2.1 Baseline CNN

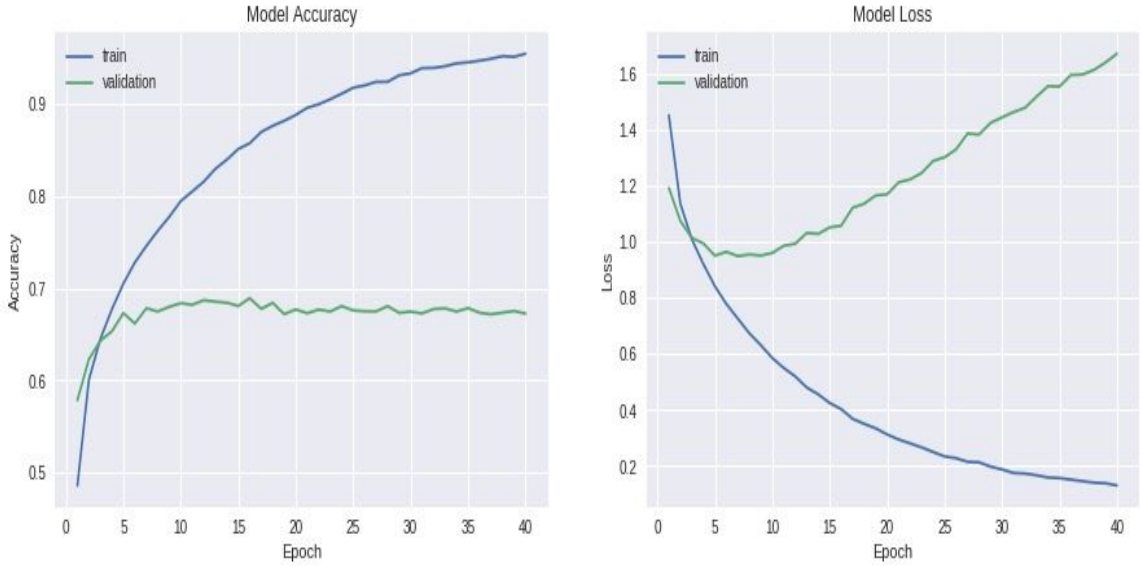
Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d_6 (MaxPooling2	(None, 16, 16, 32)	0
dropout_7 (Dropout)	(None, 16, 16, 32)	0
flatten_4 (Flatten)	(None, 8192)	0
dense_7 (Dense)	(None, 256)	2097408
dense_8 (Dense)	(None, 10)	2570

Total params: 2,100,874
Trainable params: 2,100,874
Non-trainable params: 0

None

In order to check the improvement, we establish a baseline model with convolutional neural network (CNN). Obviously, a very complex CNN always suffers from degradation. Therefore, we build a simple classical architecture with convolutions, max pooling and dropout to balance the overfitting. The test accuracy reaches 67.27% with 40

epochs and batch size 200. The accuracy and loss plots are shown as follows. Notice that there is a large gap between train and test accuracy, and the test loss starts increasing at certain point. Thus we need exploring more architectures to improve the performance.



2.2 Very deep convolutional networks (VGG)

2.2.1 VGG 11&19 Architecture

In this project, we try two different vgg architectures, VGG11 and VGG19. We will compare the differences of architecture and evaluate these two architectures in accuracy and computational efficiency.

VGG-19 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 19 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. In this case, we use vgg19 to classify images of dataset of CIFAR-10 to 10 categories. The architecture is shown as below.

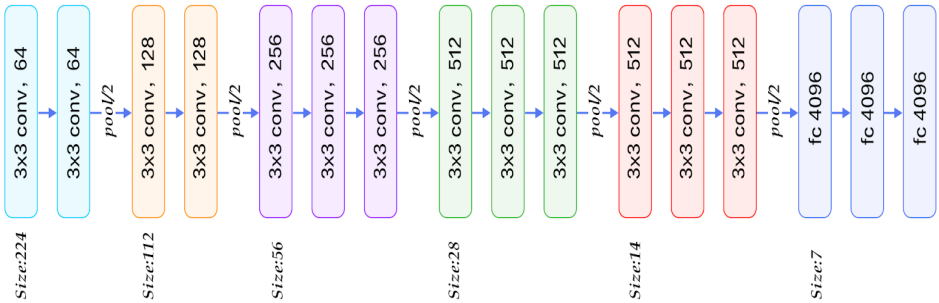


Figure 1. VGG19 architecture

VGG11 consists of 11 sequential layers. It contains seven convolutional layers, each followed by a ReLu activation function, and five max pooling operations, each reducing feature by 2. All convolutional layers have 3X3 kernels. The first convolutional layer produces 64 channels and then as the network deepens, the number of channels doubles after each max pooling operation until it reaches 512. The architecture of VGG11 is shown as below.

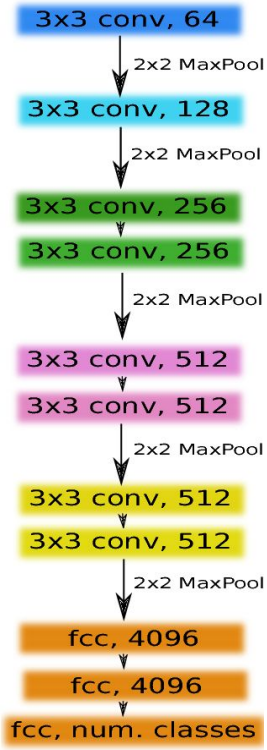


Figure 2. VGG11 architecture

2.2.2. Model performance

I. Accuracy

VGG-11

The accuracy of the network on the 10000 test images is 90.38%.

```
Accuracy of plane : 87.50 %
Accuracy of   car : 92.31 %
Accuracy of  bird : 84.62 %
Accuracy of   cat : 77.27 %
Accuracy of  deer : 92.31 %
Accuracy of   dog : 66.67 %
Accuracy of  frog : 72.22 %
Accuracy of horse : 91.67 %
Accuracy of  ship : 95.24 %
Accuracy of truck : 94.12 %
```

VGG-19

The accuracy of the network on the 10000 test images is 92.81%.

As we can see from the results above, the accuracy of VGG-11 is a little bit lower than VGG-19. The final accuracy of test dataset is 90.38%, while the accuracy of VGG-19 is

92.81%. From the table of accuracy in different categories, we find the VGG-11 performs the best in category ‘ship’.

II. Computational Efficiency

The training time of VGG-11 is shorter than VGG-19. It could result from the complexity of who convolutional neural network. VGG-19 has a deeper network and a larger total parameters, therefore, it costs longer time to train the network. However, the final accuracy of VGG-19 is higher. So, sometimes we need to trade off between the speed and accuracy.

III. Model Comparison

Compared VGG-19 with VGG-11, we find the loss of VGG-19 starts with a lower level, approximately 2.058 in the first epoch. However, the loss of VGG-11 is about 79.784. In the following epoch, the loss of VGG-11 shows a trend of fast descending. It ends with the loss of 0.2. Therefore, the final accuracy of VGG-11 and VGG-19 has no significant difference in the test dataset.

2.2.3 Summary

In the aspect of model accuracy, VGG-19 performs better. However, if considering the training cost, VGG-11 is more computational efficient. We analyze the reasons why there exists an accuracy difference between our experiment and the data presented in the report. It could result from (i) the input dataset. The size of input data in the VGG paper is 224*224*3, while the size of CIFAR-10 data is 32*32*3. The size of convolutional layer is shrinked, however, wider convolutional layer could provide more information. (ii) The set up of parameter, such as epoch, batch size, iterations. The original parameter was train based on a different dataset with larger categories and image size. (iii) The problem of overfitting. When training the model, the problem of overfitting could lead to the large gap between the training set and test set.

2.3 Network In Network(NIN)

The deep network structure Network In Network proposed by Lin, Chen, and Yao (2013) is influential and inspired the Inception line of deep architectures from Google. In convolutional neural networks (CNNs), there are convolutional layers and pooling layers. The convolutional layers take the inner product of the filter or kernel and then the outputs pass through nonlinear activation functions or pooling. The figure below shows the convolution operation.

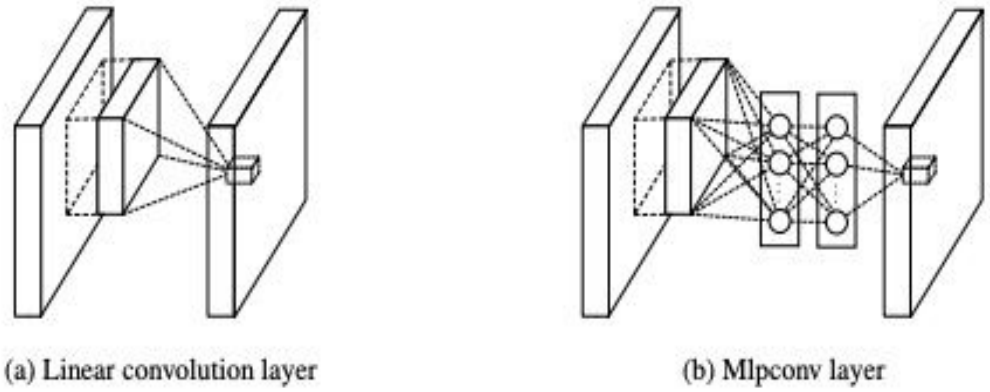


Figure 3. Comparison of linear convolution layer and mlpconv layer

Note that the convolution filters in CNN are generalized linear models, while Lin, Chen, and Yao (2013) argue that GLMs have low abstraction. The abstraction here means the feature is invariant to the variants of the same concept. The good performance of GLMs are based on the assumption that the latent concepts are linearly separable. That is the line defined by GLM keeps the variants of the concepts on the same side. However, the data for the same concept cannot separate by linear functions most of the time. So they propose a more potent nonlinear function approximator to improve the abstraction ability of the model. Like in Figure 3(b), a "micro network" structure works as the filter and multilayer perceptron (MLP) is chosen as the micro network. The paper provides new take on how the convolution filters are designed and how to map extracted features to classes. The two main components of NIN structure are the MLP convolution layers and the global average pooling layer.

2.3.1 MLP Convolution Layers

The traditional linear convolution filters extract features out of images. In the early stage, the layers extract primitive features like lines, corners and edges. The later layers build on the early layers and extract high level features like wheels, windows and door etc. These are known as latent features. Figure 4 shows the extracting process of one particular picture.

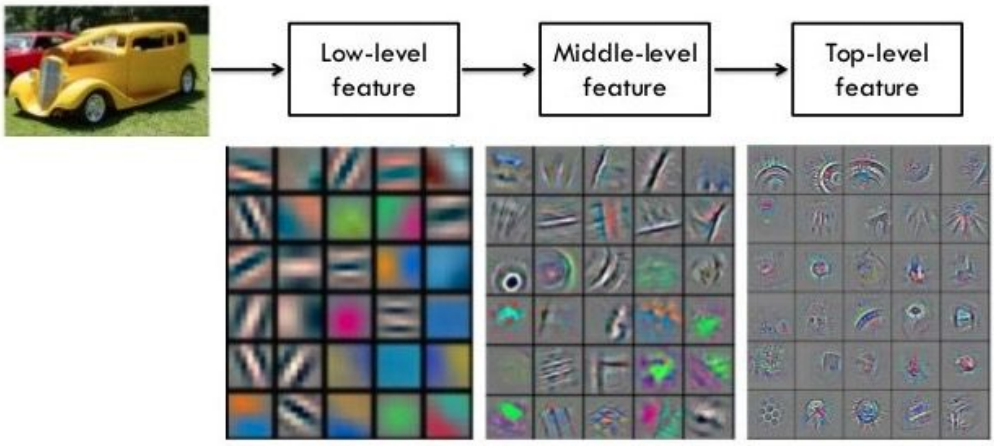


Figure 4. Extracting feature process *(Img Credit: https://arxiv.org/abs/1311.2901)*

There are lots of variations in the features so it is hard to draw straight lines to separate different types of feature or to extract features. Therefore having the MLP convolution layer is beneficial in two ways. Firstly, it is compatible with the backpropagation logic of neural nets. Secondly, MLP can be a deep model itself to separate latent features. To make it more understandable, the cross channel parametric pooling layer is equivalent to a convolution layer with 1x1 convolution kernel.

2.3.2 Global Average Pooling

In classical CNNs, the features of the last convolution layer are flattened and passed on to one or more fully connected layers, and then passed to softmax logistics layer for final class probabilities. The problem is that it is hard to interpret the category information from the convolution layers to the classifier since the fully connected layers are acting as a black box. Moreover, the fully connected layers are prone to overfitting and can be easily affected by dropout regularization. The last fully connected layer produces lots of new parameters in the network. Therefore global average pooling is applied to replace the fully connected layers in CNNs. It generates one feature map of each corresponding category in the last MLP convolution layer. Each map is averaged and the results are fed directly to a softmax layer to give final class probabilities.

One of advantages of the approach is the mapping between the extracted features and categories are more intuitive so that the feature maps can be interpreted as category confidence. Another advantage is that there is no new parameter to train, which avoids overfitting. Finally, it sums out the spatial information, thus it is more robust to spatial translations of the input.

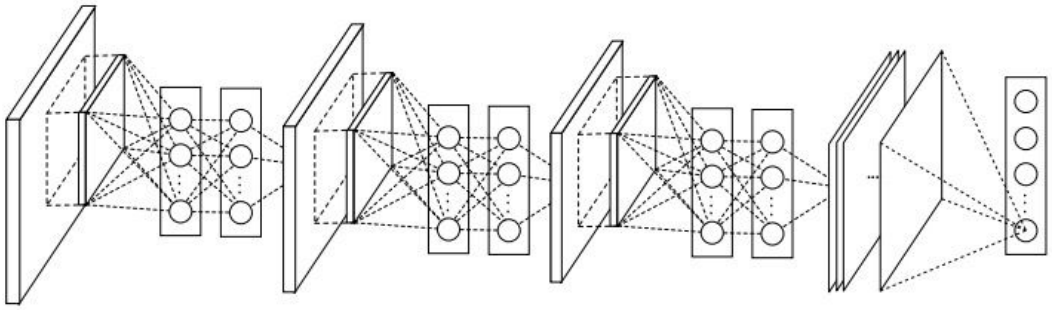
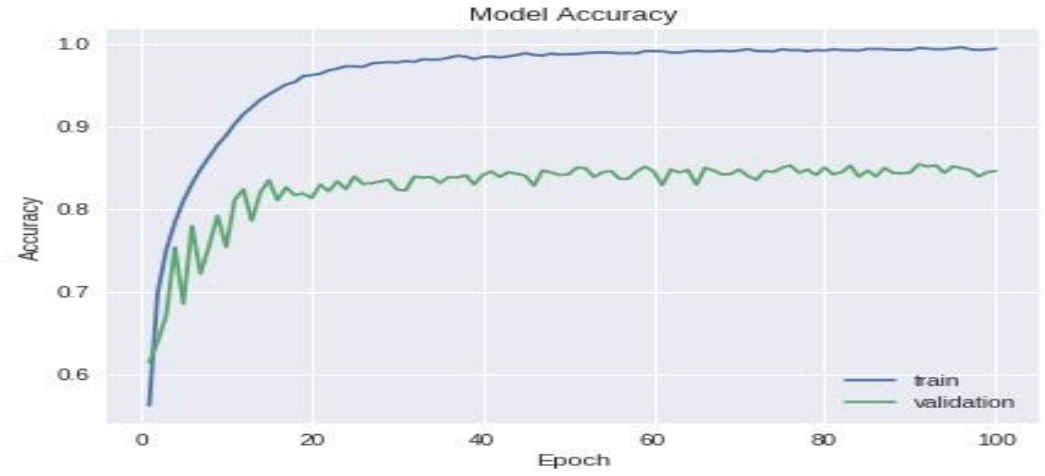
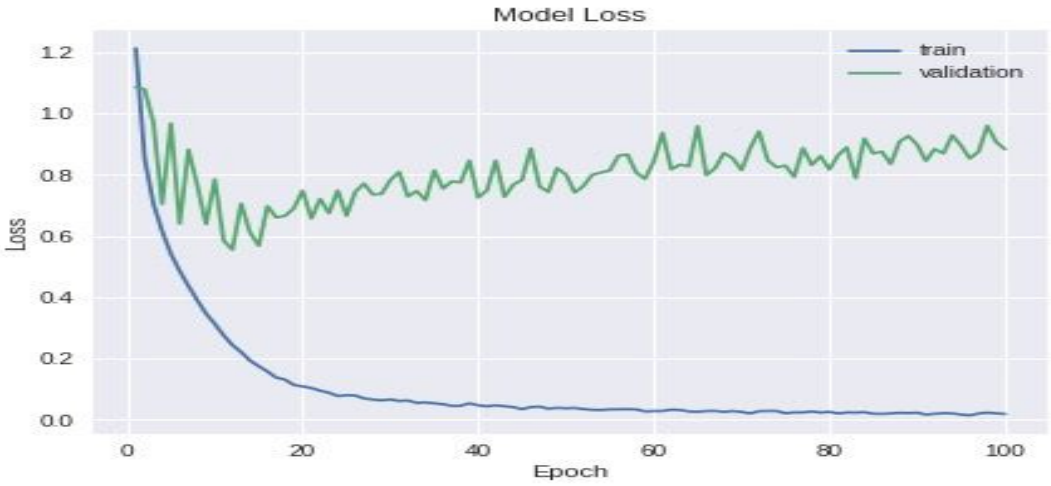


Figure 5. The overall structure of Network In Network

The NIN we use include the stacking of three MPL convolution layers and one global average pooling layer. Our NIN implementation is 2-layer with batch normalizations and we tried to use 3-layer NIN but the performance is not improved. Our final NIN model (epochs=100, batchsize=128) gives the test accuracy 84.57% and train accuracy 99.33%. The following graphs show the accuracy and loss rates and our NIN structure.





Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 32, 32, 192)	14592
batch_normalization_13 (Batc	(None, 32, 32, 192)	768
activation_13 (Activation)	(None, 32, 32, 192)	0
conv2d_14 (Conv2D)	(None, 32, 32, 160)	30880
batch_normalization_14 (Batc	(None, 32, 32, 160)	640
activation_14 (Activation)	(None, 32, 32, 160)	0
max_pooling2d_3 (MaxPooling2	(None, 10, 10, 160)	0
dropout_5 (Dropout)	(None, 10, 10, 160)	0
conv2d_15 (Conv2D)	(None, 10, 10, 192)	768192
batch_normalization_15 (Batc	(None, 10, 10, 192)	768
activation_15 (Activation)	(None, 10, 10, 192)	0
conv2d_16 (Conv2D)	(None, 10, 10, 256)	49408
batch_normalization_16 (Batc	(None, 10, 10, 256)	1024
activation_16 (Activation)	(None, 10, 10, 256)	0
average_pooling2d_3 (Average	(None, 3, 3, 256)	0
dropout_6 (Dropout)	(None, 3, 3, 256)	0
conv2d_17 (Conv2D)	(None, 3, 3, 192)	442560
batch_normalization_17 (Batc	(None, 3, 3, 192)	768
activation_17 (Activation)	(None, 3, 3, 192)	0
conv2d_18 (Conv2D)	(None, 3, 3, 192)	37056
batch_normalization_18 (Batc	(None, 3, 3, 192)	768
activation_18 (Activation)	(None, 3, 3, 192)	0
global_average_pooling2d_3 (	(None, 192)	0
dense_2 (Dense)	(None, 10)	1930

Total params: 1,349,354
Trainable params: 1,346,986
Non-trainable params: 2,368

2.4 Wide Residual Network(WRN)

The traditional deep residual networks performs well in image recognition tasks. However, it usually costs doubling of layers with a small accuracy improvement and hence the feature usages will diminish. Sergey Zagoruyko and Nikos Komodakis propose the “wide residual networks” , which decreases the depth and increases the width (k) of the residual networks. With reference to their work, we focus on wide residual networks(WRN) with 3x3 convolutions for all the models we trained, as they will shorten the training time and have a relatively high training accuracy compared with other type of convolutions in residual blocks. The structure of WRN is shown below, “k” represents the width of the WRN. When k=1, the structure represents an ordinary residual network.

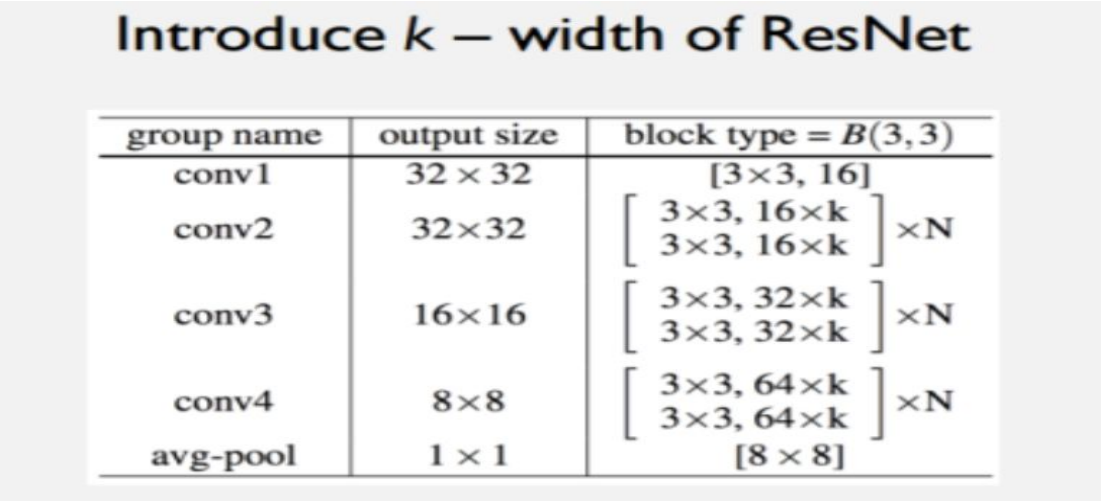


Figure 6. The structure of WRN

2.4.1 Architecture

We train the model “WRN-16-4” and “ResNet-40” (depth=40,k=1) as they have similar number of parameters (~2.8M,~2M). The objective is to find out whether “WRN-16-4” will outperform “ResNet-40”, either in testing accuracy or the training time. The result will indicate whether increasing the width of the block will decrease the training time and enhance the accuracy. The architectures for “WRN-16-4” and “ResNet-40” are shown below.

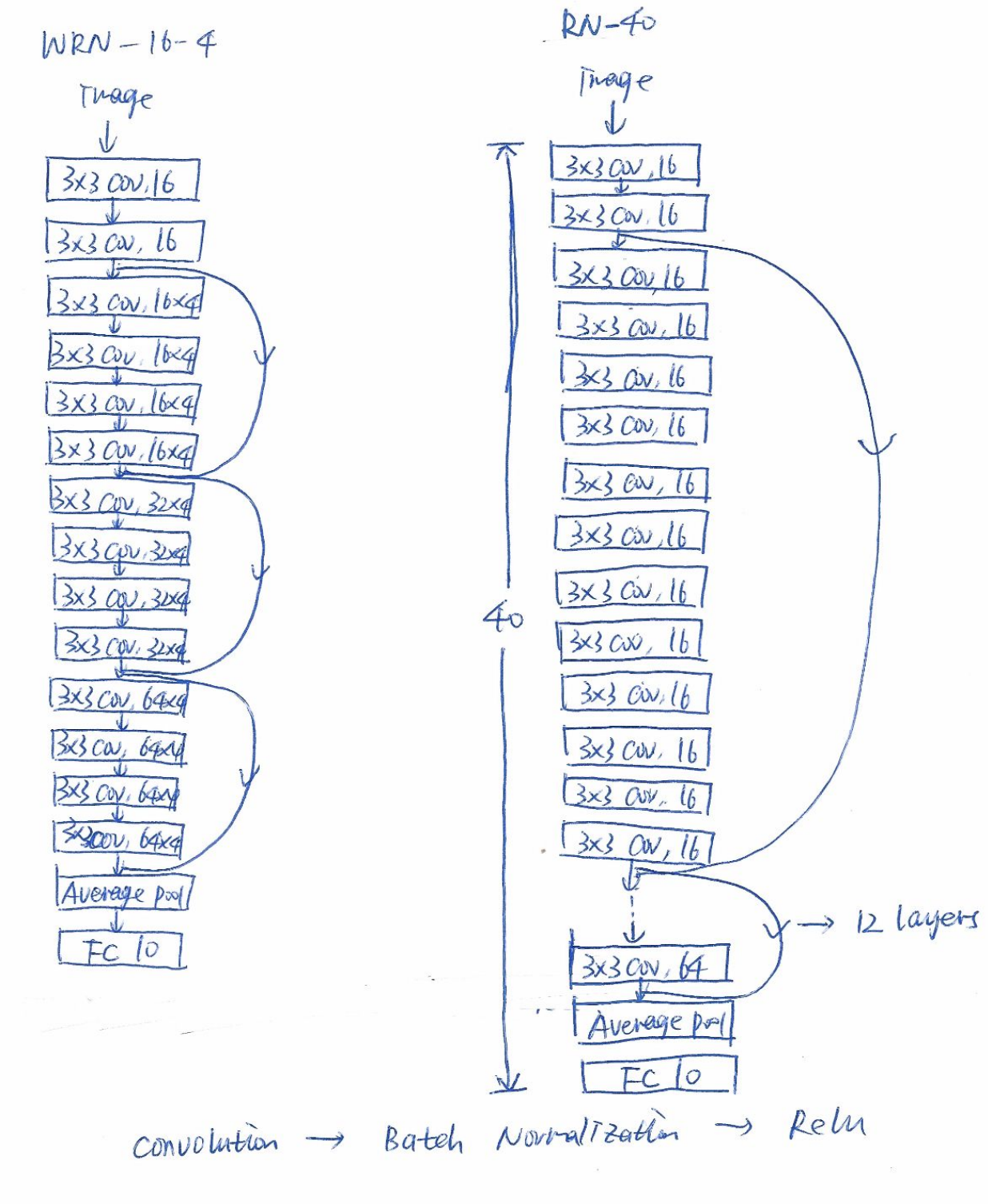
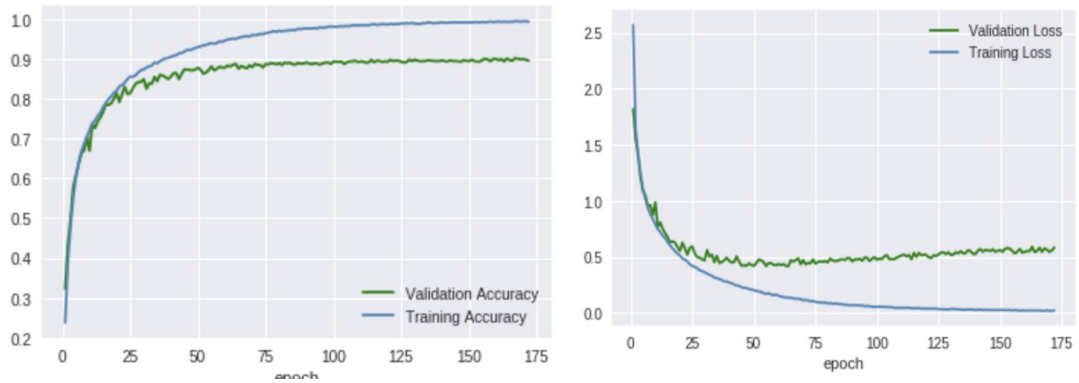


Figure 7. The architectures of "WRN-16-4" and "ResNet-40"

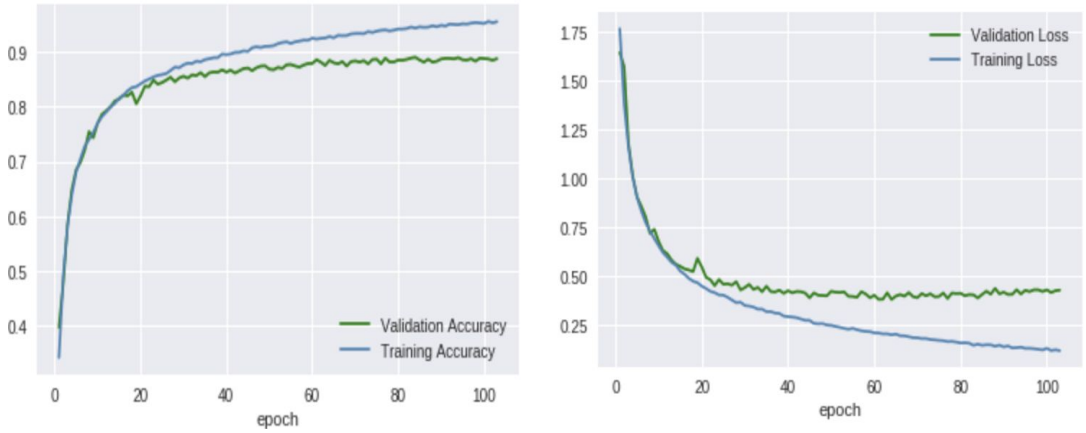
2.4.2 Model Performance

For "WRN-16-4":



As we can see after 125 epochs, the accuracy becomes stable, but the loss, especially for the validation dataset shows an upward trend and validation loss << training loss, which shows some over-fitting. This may caused by lack of L1/L2 regularization and complete data augmentation, as we only tried the horizontal flips as the data augmentation. The testing accuracy is 89.4% and the training time is about six hours for 150 epochs (batch-size=120).

For "ResNet-40":



The training accuracy still shows an upward trend after epoch 100, which is different from "WRN-16-4", a flatter end. This tells us "ResNet-40" shows a more serious over-fitting problems than "WRN-16-4", which may be explained by its deeper layers as the pre-processing of the data are the same in two models. The testing accuracy is 88.2% and the training time is about six hours for 100 epochs (batch-size=120) .

Comparison:

The training time for "WRN-16-4" is shorter than "ResNet-40" , which is related to the model structure. Although the number of parameters has a linear increasing trend with the

layers and a quadratic increasing trend with the widening factor, GPU is more efficient in parallel computation. Hence the training timing cost for "WRN-16-4" is less than the "ResNet-40". Moreover, the testing accuracy of "WRN-16-4" is a little bit higher than "ResNet-40", this can be explained by the diminishing feature usage with increasing of the neural network layers. Therefore, the "WRN-16-4" outperforms "ResNet-40" in our experiment.

3. Conclusion and future work

From all models we've trained, we obtain the following conclusions:

(1) There is always a tradeoff between computation efficiency and the accuracy. Although increasing the number of layers will improve the accuracy, it may also increase the computation time.

(2) For residual networks with comparative number of parameters, decreasing the number of layers and increasing the width of the blocks may enhance the computation efficiency and test accuracy.

(3) The validation accuracy and loss have more fluctuations compared with training accuracy and loss. This may be caused by the biased validation dataset and the non-regularized training process.

(4) In NIN architecture, after applying the batch normalization in each channel, the classification performance has been improved. Dropout is used between each NIN blocks for regularization and hence no additional regularizations are needed.

Future work:

(1) Overfitting is a problem in our models(either slightly or severely), we may try to use the complete data augmentation tools (horizontal and vertical flips, rotation, zoom, color-purtabation) in data pre-processing to see if it works. In addition, we will apply the L2-regularization for model weights.

(2) Because of the expensive time-consuming problem for training process, we didn't try the wide residual network with different width factor k. However, we are interested in whether there is an upper bound for the best accuracy as we increase the width of the blocks, which could be part of our further study.

References

1. Min Lin, Qiang Chen, Shuicheng Yan. *Network in network.*

https://arxiv.org/pdf/1312.4400.pdf

2. Sergey Zagoruyko and Nikos Komodakis: Wide Residual Networks. Retrieved from https://arxiv.org/pdf/1605.07146v1.pdf

3. TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation. Retrieved from

https://arxiv.org/pdf/1801.05746.pdf

4. Very Deep Convolutional Network For Large-Scale Image Recognition

https://arxiv.org/abs/1409.1556

5. Anand Saha, Network in network architecture: The beginning of Inception.

http://teleported.in/posts/network-in-network/

6. Kaggle

https://www.kaggle.com/bingdiaoxiaomao/network-in-network-nin-with-keras