USC Viterbi
School of Engineering

DEPARTMENT
Electrical and Computer Engineering

EE 569
Homework #5:
CNN Training on LeNet-5

Name: Zhiwei Deng
Date: 04/03/2020

Contents

1. Problem 1

1.1 CNN Architecture

1.1.1 CNN Components Description

The fully connected layers: The fully connected layers (FC layers) work as a classifier in the Convolutional Neural Networks. The features that extracted by the convolutional layers are the input of the FC layers. The input features are mapped from the feature space into another feature space through the FC layer. And eventually map the features into label/output spaces. The basic mathematical operation in FC layers are matrix multiplications. The outputs of every FC layer are computed determined by the weight matrix and the bias vector. Formula is shown as equation (1).

$$\boldsymbol{Y_j} = \sum_{\boldsymbol{i}} \boldsymbol{X_i} * \boldsymbol{W_i} + \boldsymbol{b_i} \tag{1}$$

To get a better performance, a back-propagation procedure is used to optimize the parameters in weight matrices and bias vectors. This process is done by computing the derivatives of the loss function to the X, Y, W and b, trying to find a global (or local) minimum of the loss function. The computing formulas are shown as below in equation (2-4). In the equations, L is loss function, W is weight matrix, X is the input matrix, Y is the output matrix. The L's derivative of Y can be computed directly (Jacobian matrix).

$$\frac{\boldsymbol{\partial L}}{\boldsymbol{\partial X}} = \frac{\boldsymbol{\partial L}}{\boldsymbol{\partial Y}} * \boldsymbol{W^T} \tag{2}$$

$$\frac{\boldsymbol{\partial L}}{\boldsymbol{\partial W}} = \boldsymbol{X^T} * \frac{\boldsymbol{\partial L}}{\boldsymbol{\partial Y}} \tag{3}$$

$$\frac{\boldsymbol{\partial L}}{\boldsymbol{\partial B}} = \boldsymbol{1^T} * \frac{\boldsymbol{\partial L}}{\boldsymbol{\partial Y}} \tag{4}$$

The convolutional layers: The convolutional layers work as feature extractors in the Convolutional Neural Networks. It uses filters to extract the different local features of the input image. A filter/mask is used to calculate the corresponding features of the image's local region. The filtered image is computed by correlated the pixel value with the mask's weights, which can be tuned by the feed-forward process. The convolution operation is shown as below in Figure 1. Generally, the more filters we use, the more features we can get and more precise result we can get. There are 3 hyperparameters we can tune, depth of the volume, stride and padding. The depth of the volume can be determined by the number of the filters. The stride is used to prevent the filters to overlap and improve the efficiency. The padding style is used to make sure the marginal pixels' information can be obtained. Also, the input images might have multiple channels and the convolutional layers may have different multiple filters for these channels.

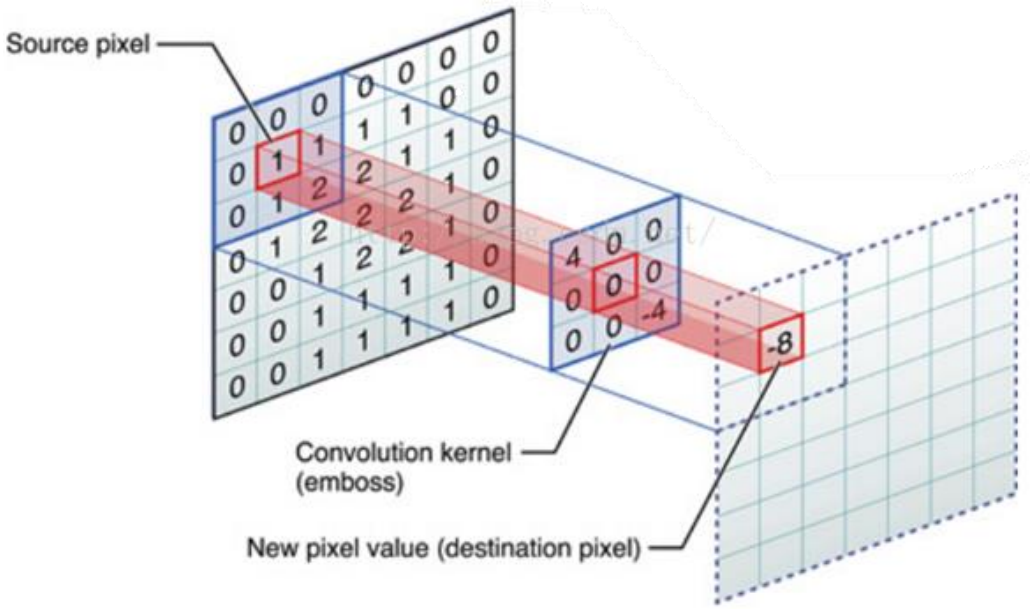

Figure 1. Convolution Operation

The max pooling layer: The main purpose of the pooling layers is to down sample the input image to reduce the redundant information, suppress the feature data and simplify the network complicity. While the dimension of the data reduces, the probability of overfitting of the network decreases. Also, the pooling layer increase the speed of the computation since the data is down sampled. In some level, it can also realize the invariance of scaling, rotation and translation. The max pooling layer is one kind of the pooling layers. It picks the largest value of a certain region as the pooled value. In general, the max pooling window does not overlap with each other and the whole process of the max pooling is shown as Figure 2.

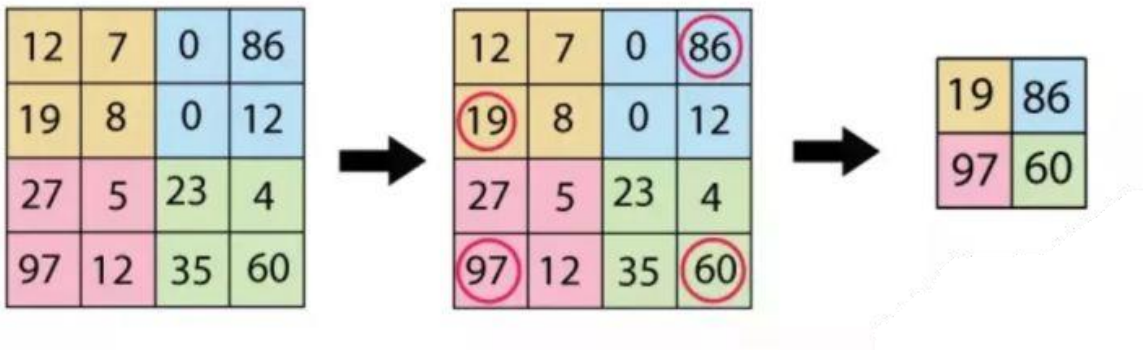

Figure 2. Max Pooling Operation with Window 2 by 2

The activation function: To simulate the functions of the neurons, since every neuron is activated when it is working. Then, every node of the hidden layers and output layer should have an activation function activate the certain nodes to work. The main purpose of the activation function is to introduce the non-linear component into the network. Since in each layer, if we only apply the weight matrix and the bias vector, the outputs of each layer can only be the linear combinations of the inputs as y = w*x + b. In this case,

the CNN becomes a simple perceptron, which cannot discriminant non-linear characteristics in the data as shown in Figure 3. So, the activation function is used to break the linearity between layers and introduce the non-linear property into the whole network.

There are some widely used activation function. Sigmoid function reduces the impact of the negative outputs and limit the output the between 0 and 1. ReLU function discard all negative responses and keep the positive responses as they are. For more specific, ReLU function can avoid the gradient decrease or gradient disappearance. In conclusion, activation function is one of the most important reason why CNN can work so well on the non-linear problems.

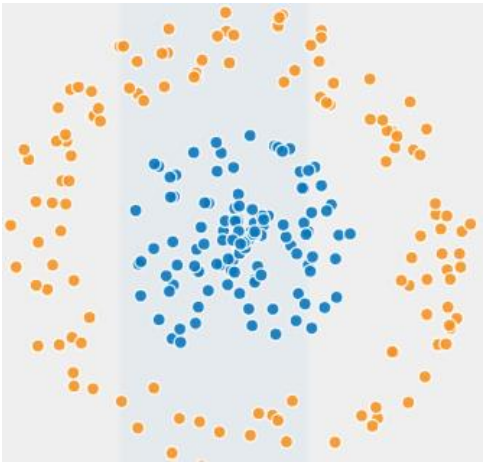

Figure 3. Non-Linear Discriminate Data

The softmax function: Softmax function acts like a classifier in the end of the CNN. It converts the output values of each classes to the possibilities of each classes between 0 and 1. Firstly, it uses exponential function to map the output values onto the positive numbers of the real line. Then it normalizes these values to turn the outputs into possibilities. The

corresponding class of each data is assigned to the class that possess largest value. The formula of softmax function is shown as equation (5).

$$\boldsymbol{p(y)} = \frac{\boldsymbol{exp(Wy*x)}}{\sum_{\boldsymbol{c}} \boldsymbol{exp(Wc*x)}} \tag{5}$$

1.1.2 Over-fitting Issue

Over-fitting means the CNN model matches the training data too well and decrease its own possibility to generalize. In an over-fitted model, training accuracy usually is much higher than the testing set's accuracy, since in some level the model becomes an ad-hoc network which works well for the training set and lost its ability to generalize on other data. With smaller training set and too many epochs, the model is more likely to over-fit. An example is shown as Figure 4 about the under-fit, good-fit and over-fit model.

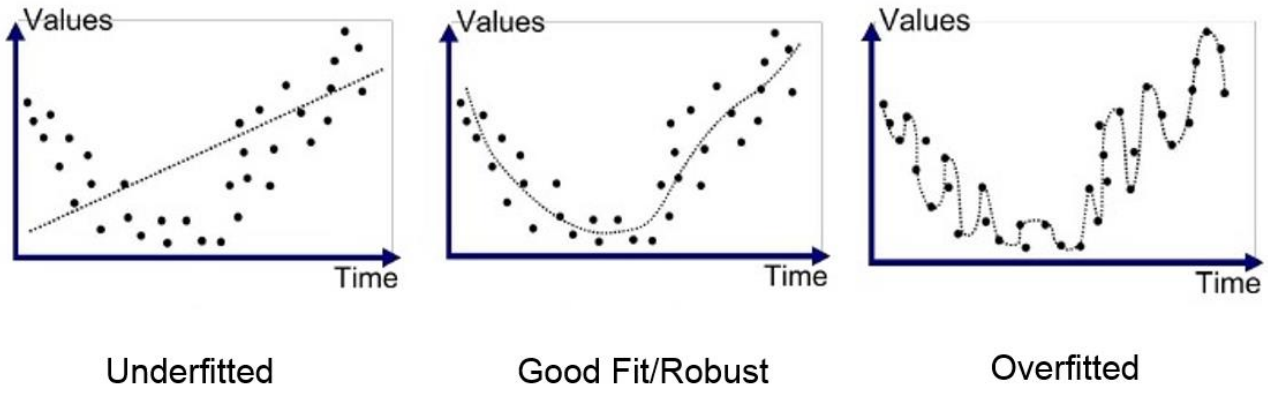

Figure 4. Examples of Data Fitting

Techniques to prevent the occurrence of over-fitting are various. Firstly, we can use **data augmentation** to prevent the over-fitting from the source. This technique generates more data using the current data set by doing horizontal flip or certain degree rotation. This technique works because it gets

a more generalized dataset, which contains more ground truth image data. So, it increases the generalization ability of the model and decreases the possibility of over-fitting. From the training prospect, we can **stop the training process early** to prevent the occurrence of over-fitting. This technique works because of the error of test sets or validation sets is normally decrease at first while the number of epochs increases. Then the error of test sets or validation sets is going to increase due to the over-fitting occurrence. In this case, stopping the training process early to avoid the over-fitting makes sense. **Regularization** is another technique to prevent the over-fitting. This technique prevents over-fitting by introducing the magnitudes of the parameters into the loss function. Since for a model which has parameters with large scale, the result will change drastically even the data shifts slightly, and the generalization ability is poor. By introducing the magnitude of parameters, the magnitude of parameters will shrink along with the cost function is optimized, which indicate a higher generalization ability. **Dropout** can also be used to introduce randomness and system sparsity into the model to avoid the over-fitting occurrence.

1.1.3 Why CNN works better?

There are several reasons for CNN's excellence in CV problems. For instance, in the image classification tasks. Firstly, different from traditional methods, which only use the single image's information, since CNN is data-driven algorithm, it uses datasets which contains much more information than a single image and might get more information under the surface. Secondly,

CNN is a supervised learning algorithm, the parameters can be tuned by using the back-propagation process. From this prospect, CNN can get even more information about the data since it is a label-based learning. Thirdly, CNN has a feedforward scheme, as known as the back-propagation process. This kind of process makes the CNN has some robustness, since every parameter can be tuned by the feedback of the results. Fourthly, there are many parameters in CNN that can promise most image of the training sets are labeled correctly if enough epochs given as the over-fitting stated. So, it can produce a more precise non-linear subspace than traditional process, which can distinguish images with each other. Fifthly, the model for the image classification cannot be used to object detection, while both of them might needs to do the edge detection process. The CNN models mostly are task-based. Unlike Canny detector or other filters, one CNN model usually cannot be applied on other tasks. That is why it can beat many traditional methods, since it focus on a single task. Lastly, it uses much more computation resources. In general, with more time and resources it used, the result might be more precise. The CNN only can perform well when given a plenty of time for training and will not have good results if time is limited.

In conclusion, the reasons for the excellence of CNN are more computational resources, single task specified, label-based supervise learning, data/statistics-driven, BP process and sufficient parameters.

1.1.4 Loss Function and BP Optimization Procedure

Loss function: Loss function is a function that used to measure the

difference between predicted results and ground truth results, expressed as L(Y, f(x)). In classification problems, the mostly used loss function is Cross Entrophy loss function. The formula is shown as below in equation (6), where N is the total number of samples, M is the number of classes, y represent whether the prediction right(1) or not(0), p means ith sample's probability for c class. Since the error rate cannot represent the model performace, and the learning process would be very slow if mean squared error(MSE) used. So in classification problems, cross entrophy is used. To get better model parameters, we can try to minimize the loss function based on the training data. Loss function is very important to CNN, finding the global minimum of loss function is the final goal of whole CNN procedure.

$$\mathbf{L} = \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i}}^{\boldsymbol{N}} \boldsymbol{L_i} = -\frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i}}^{\boldsymbol{N}}\sum_{\boldsymbol{c}}^{\boldsymbol{M}} \boldsymbol{y_{ic}}\mathbf{log}\,(\boldsymbol{p_{ic}}) \tag{6}$$

Back Propagation Procedure: Backpropagation is a way of computing loss function's gradients of weights and bias, then update the weight and bias parameters to reduce the loss of the model to the minimum. The gradient's orientation represents the direction of the max change rate and if we use this steepest to do the descent. So, we need to use a gradient descent method to find the certain global minimum. So, formula as equation (7) is used to make sure the loss function has a negative change, which means the increase of the accuracy.

$$\mathbf{w}' = \boldsymbol{w} - \boldsymbol{\alpha}\,\frac{\boldsymbol{\partial L}}{\boldsymbol{\partial w}} \tag{7}$$

$$\boldsymbol{b}' = \boldsymbol{b} - \boldsymbol{\alpha}\,\frac{\boldsymbol{\partial L}}{\boldsymbol{\partial b}} \tag{8}$$

Also, the error of the output layer is going to be propagated backward. Firstly, we need to forward propagation to calculate the output results. The equations are used as equation (9-10). Where o represents the output result and a represent the output of the previous layer, σ means the activation function. Then at the last layer, the output layer, the error of the results for each node of the last layer is computed using the loss function's gradient by equation (11). At last, the BP method propagates this error to the previous layers to calculate the error that happens in the nodes of each layer. The calculation is shown as equation (12). After the error of each layer is propagated and computed, it provides a more specific and precise modification for the weight and bias to improve the performance of the network. The gradient of the weights and bias are shown as equation (13-14), we can use this equation to update the weights and bias of every node. As the procedure stated, the error of outputs can propagate through the network and train the parameters by using the gradient descent method above.

$$\boldsymbol{r^l = w^l * a^{l-1} + b^l} \tag{9}$$

$$\boldsymbol{o^l = \sigma(r^l)} \tag{10}$$

$$\boldsymbol{\delta^l = \frac{\partial L}{\partial r^l} = \frac{\partial L}{\partial o^l} * \frac{\partial o^l}{\partial r^l} = \frac{\partial L}{\partial o^l} * \sigma'(r^l)} \tag{11}$$

$$\boldsymbol{\delta^l = (w^{l+1})^T \delta^{l+1} * \sigma'(r^l)} \tag{12}$$

$$\boldsymbol{\frac{\partial L}{\partial w^l} = a^{l-1} * \delta^l} \tag{13}$$

$$\boldsymbol{\frac{\partial L}{\partial b^l} = \delta^l} \tag{14}$$

Another main method that back-propagation works is the chain rule of the derivative's computation. More specifically, let y be the output result and x be the input of the network. Evey input of each hidden layer is output of the previous layer. So, the function between x and y is shown as equation (15). For the same reason, we can get the relation between weights, bias and the loss function can be represented by equation (16). So, loss function's gradient of w can be expressed as equation (17). This equation is called the chain rule of the derivative. This chain rule can also be seen in equation (11). The chain rule can speed up the whole CNN algorithm, since according to equation (12), the gradient and errors of each layer can be computed using the gradient and error results of previous layer. Also, the loss function's gradient of certain layer can be computed only using the result of previous layer and the errors of same layer. So, the back-propagation process saves much computation compared with the forward-propagation computation.

$$\boldsymbol{y = h_l(h_{l-1}(\dots h_1(x)))} \tag{15}$$

$$\boldsymbol{z = f(x,\ w,\ b)\quad a = g(z)\quad y = s(a)\ \ L = t(y, y')} \tag{16}$$

$$\boldsymbol{\frac{\partial L}{\partial \mathrm{w}} = \frac{\partial L}{\partial \mathrm{y}} * \frac{\partial y}{\partial \mathrm{a}} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial \mathrm{w}}} \tag{17}$$

1.2 CIFAR-10 Classification

1.2.1 Abstract and Motivation

Convolutional neural network has changed the field of computer vision and image processing approaches significantly since it outperforms many traditional algorithms on the aspect of accuracy in many problems like object recognition, image classification and image matching. Also, CNNs are widely used in many industries on visual tasks like Facebook.

In this part, I implemented the classical LeNet-5 CNN to classify the CIFAR-10 dataset. The network architecture has 2 convolutional layers, 2 pooling layers and 2 FC layers. Then 5 different sets of parameters are excuted to compare with each other. The comparison and the observations are given and discussed. At last, some advanced techniques are used and every parameters are fined tuned trying to find the best-performed parameters.

Note: For the first sub-problem, the network architecture is not changed. The only changed parameters are learning rate, epochs and batch size. For the second sub problem, the number of the filters of each convolutional layers are changed and the dropout techinique is used.

1.2.2 Approach and Procedures

CIFAR-10: CIFAR-10 is a dataset which contains 50000 32 by 32 colofurl images as training set and 10000 similar images as testing set. All images can be classified into 10 classes.

LeNet-5: LeNet-5 is a basic convolutional neural network which only contains 2 convolutional layers. It was designed to recognize the handwritten

digits dataset -- MNIST, which contains 28 by 28 grayscale images. Also, this recognization can be seen as a ten-class classification problem. From the introduction of CIFAR-10 above, we can see that CIFAR-10 shares some similarities with MNIST. They both have 10 classes and the scale of the images of both sets are similar. So, it might be reasonable to use LeNet-5 to classify CIFAR-10. However, there are some differences in CIFAR-10 that makes LeNet-5 cannot be the perfect tool to solve this problem. The images in CIFAR-10 are colorful tha have 3 channels, which is much more complicate compared to MNIST and make LeNet-5 not good enough to achieve a high accuracy.

Libraries and Implementation: In this problem, I used Jupyter Notebook as IDE, which makes the real-time excution possible and has a better GUI. For CNN construction, I used Keras as the OpenSource API, which is using TensorFlow as the backend. For the implementation, to keep the same architecture of the network as stated in the assignment, the parameters I tuned in the first sub-problem are only learning rate, epoch number and batch size. In the second sub-problem, I changed the number of filters but I kept the ratio of the number of filters in two convolutional layers as 3/8. For both problems, no data augmentation is used. Also, the Adam optimizer and train data shuffle are used for both problems.

Note: Since 5 sets is a small amount to try many parameters in practice. So, some parameters that I do not include in the report will still be discussed in the discuss section.

1.2.3 Experimental Result.

The result graphs of 5 sets different parameters are shown as below in Figure 5 to 9, parameters are as (learning rate, batch size, epoch numbers).

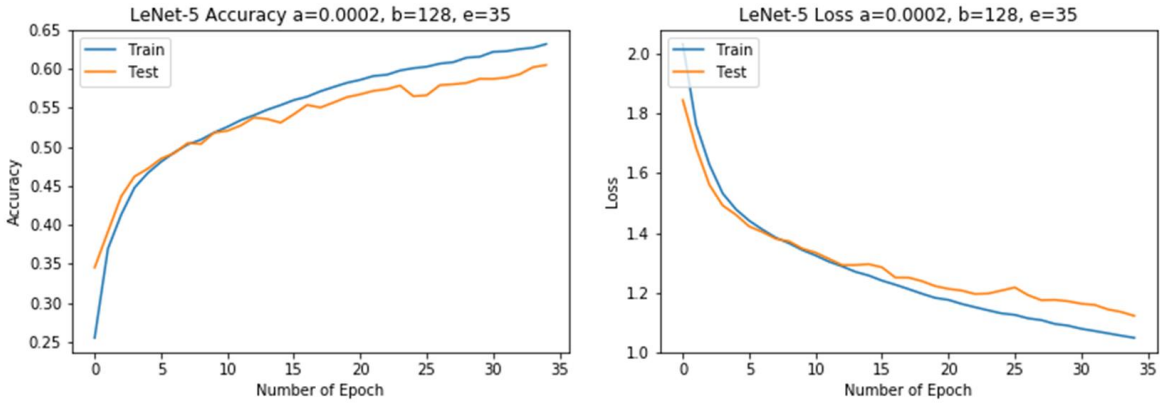

Figure 5. Accuracy and Loss to Epochs (0.0002, 128, 35)-7s/epoch

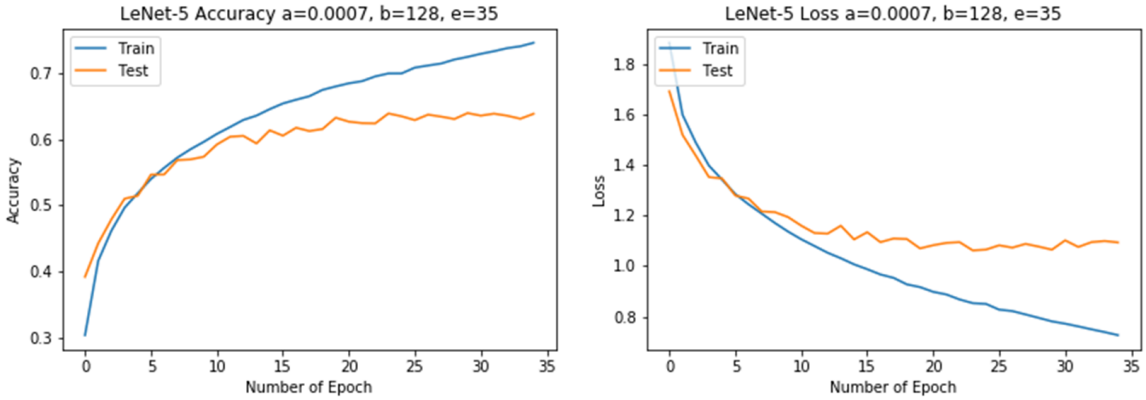

Figure 6. Accuracy and Loss to Epochs (0.0007, 128, 35)-7s/epoch

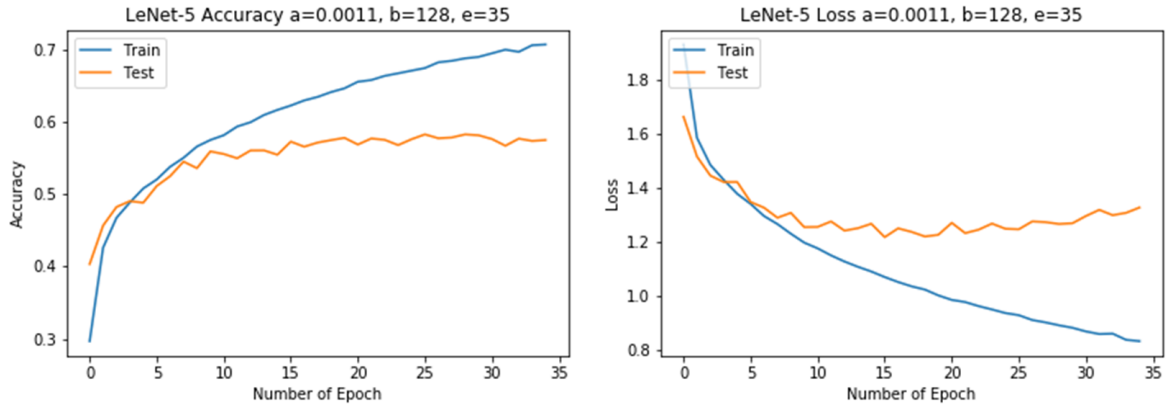

Figure 7. Accuracy and Loss to Epochs (0.0011, 128, 35)-7s/epoch

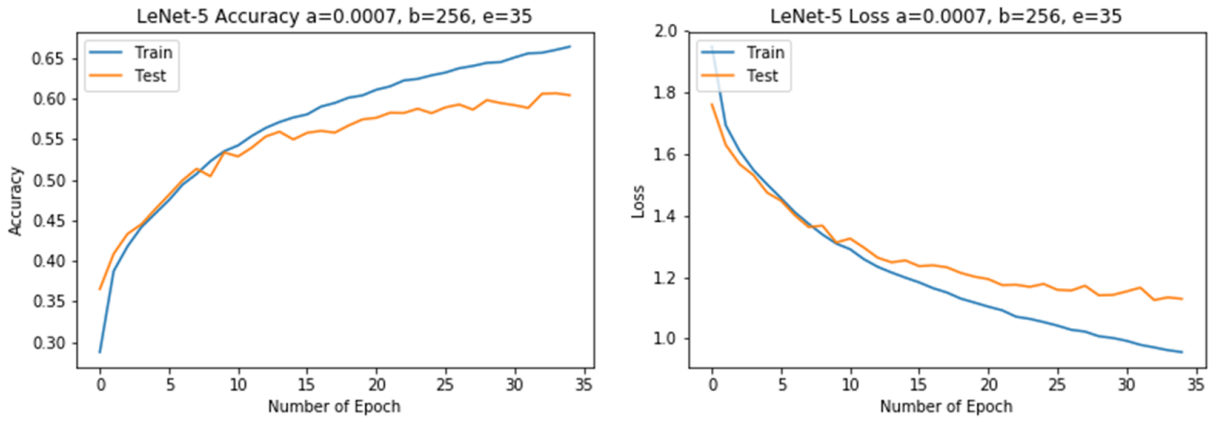

Figure 8. Accuracy and Loss to Epochs (0.0007, 256, 35)-6s /epoch

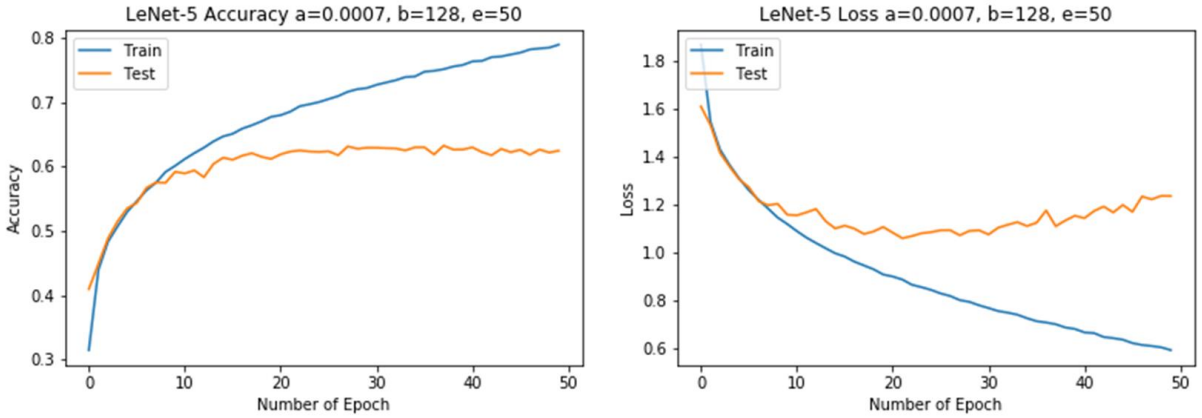

Figure 9. Accuracy and Loss to Epochs (0.0007, 128, 50) -7s/epoch

The result of the best performed parameters are shown as Figure 10.

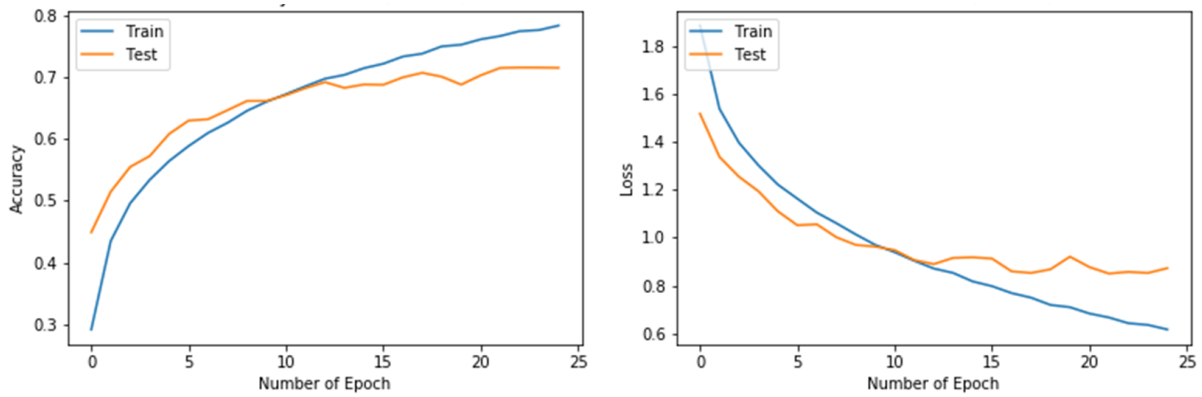

Figure 10. Best Accuracy and Loss to Epochs (0.0005, 128, 35) -7s/epoch

Filter size = (24, 64); Dropout = (0.33,0.33)

The specific performances value of each graph on training dataset and testing dataset are shown as Table 1.

Table 1. Performances of Different Settings

Filter Size	Parameters	Train Acc	Train Loss	Test Acc	Test Loss	Time	Best
(6,16)	(0.0002, 128, 35)	0.6317	1.0492	0.6049	1.1227	7s/epoch	
(6,16)	(0.0007, 128, 35)	0.7458	0.7265	0.6383	1.0936	7s/epoch	
(6,16)	(0.0011, 128, 35)	0.7068	0.8327	0.5747	1.3264	7s/epoch	
(6,16)	(0.0007, 256, 35)	0.6643	0.9563	0.6045	1.1294	6s/epoch	
(6,16)	(0.0007, 128, 50)	0.7896	0.5955	0.6246	1.2363	7s/epoch	
(24,64)+Drop	(0.0005, 128, 35)	0.7992	0.6137	0.7155	0.8727	7s/epoch	Yes

1.2.4 Discussion

Learning Rate: From Figure 5-7, we can see that when the learning rate is small, the train accuracy and test accuracy are both low but they are similar. This kind of acuracies are the characteristic of under-fitting, which means the model doesnot match the training data enough and cannot classify the test data into correct classes. This due to under too small learning rate, the weights and bias can only be tuned towards the correct direction with small magnitudes. So, with same number of epochs, the model is not trained enough. Through the loss graph, we can see that in Figure 5, the loss of the test is still decreasing, which means the model still has space to improve. While in Figure 6, both taining accuracy and testing accuracy increased. From the loss graph, we can see that the loss of the testing set becomes flatten, which means the model can fit the testing data in a proper level and it is hard for the model to decrease the testing loss. In Figure 7, the loss function of the testing set is decreased first and increased at last, which means the model has

been over-fitted. Also, from Table 1, we can see that the training accuracy increased a little but the testing accuracy decreased, which means the model over-fitted.

Mathmatically, when the learning rate is too small, it takes longer time for the model to converge to the global minimum since the step is small. When the learning rate is too large, the model might miss the gobal minimum and wander around the minimum point, which might take longer time to converge and make the model unstable. Also, when the learning rate is too large, the loss might take bigger step and shoot faster and make the model over-fit easily.

Batch Size: Batch size effects the training process a lot, when the batch size is small. The same training set is going to be divided into many small subsets to train, which will take longer time to run over an epoch. When the batch size is 256, the running time is 6s per epoch, when the batch size is 128, the running time is 7s per epoch. With a smaller batch size, the training process can converge faster, which can be seen that the training accuracy increases faster. And the model might be more easily to be over-fitted with smaller batch size.

Epoch: One epoch means a whole datasets is all trained, so the number of epochs means the iteration times of the training set. The epoch number is important for training. From the Figure 9, we can see that the training accuracy is higher compared with Figure 6, but the testing accuracy is lower. Which means the model overfits the training data and lose some

generalization ability. Also, if the epoch number is too small, the traning process might not iterate enough and cause under-fit problems. So, a suitable number of epochs can improve the accuracy of the training set and preserve the generalization ability.

Filter Size: To find the best performed parameters as shown in Table 1, I increased the filter size of each convolutional layer of the network. In general, more filters mean better performance since the filters work as the feature extraxtors and more filters means that we can get more features from the convolutional layer. However, the training time can be increased by the increasing of the filter numbers. Too few filters might make the network cannot train a effective model to solve tasks, and too many filters might make the training process takes long time to converge. So, a proper number of filters should be a trade-off for the training efficiency.

Dropout: Dropout is a technique that used to avoid the overfitting to make the model more robust and have higher generalization ability. Dropout breaks the correlations between some neural nodes, make the network less sensitive to some features or certain regions. The weights can be updated without certain hidden nodes with Dropout to prevent some features are useful only other certain conditions are satisfied. With this process, the model can train more epochs and not over-fit, and the generalization on the testing sets are better.

1.3 State-of-the-Art CIFAR-10 Classification

1.3.1 Description of Author's Work

In this part, I choose the best performed paper to discuss. The paper is *'Fractional Max-Pooling'* by Benjamin Graham of University of Warwick[2]. I will discuss the author's idea and work to get such performance in 4 parts, problem setting and goal, method description, implementation and analysis.

Problem Setting and Goal: The paper propsed a new method to improve the CNN's performance by changing the style of pooling layers. Many methods like dropout, dropconnect and small filters are proposed to improve the CNN's perforamce by improving the convolutional layers. But the pooling operation might be neglected. The goal of this paper is to propose a new pooling method to improve CNN's performance on the image and object classification. Max-pooling with window size of 2 by 2 is widely used in the many CNN networks. This pooling method is very fast and can reduce the hidden layer size quickly, and it can also introduce some level of translation and elastic distortion. But the networks are supposed to be really deep and the generalization ability is poor due to the pooling areas are disjoint. Some techniques like overlapping pooling and stochastic pooling are proposed, but the down-sampling rate is still a factor of 2. So, the author tries to make the image can be down sampled by a fractional factor to see the image in a right scale, which is presented as 'Fractional Max-Pooling' (FMP).

Method Description: The main purpose of the method is to reduce the image spatial size by a fractional factor α instead of the factor of 2 as normal

max-pooling, where 1 < α < 2. The scale of the factor is decided by random and the randomness is different with stochastic pooling. The FMP decides the pooling region randomly, while the stochastic pooling chooses values randomly in a fixed pooling region.

Let the N be the input image size, and M be the output image size. The pooling ratio α is defined as N/M. We can easily see that α determines the factor of down sampling. For normal max-pooling, the α = 2, the pooling region the disjoint 2 by 2 squares. But in FMP, the pooling regions size is going to be determined by random. For any 1< α < 2, the height and width can both chosen to be 1 or 2. For any 2< α < 3, the height and width can both chosen to be 2 or 3. The paper shows several possible pooling region division by disjoint pseudorandom FMP, which is shown in Figure 11[2].

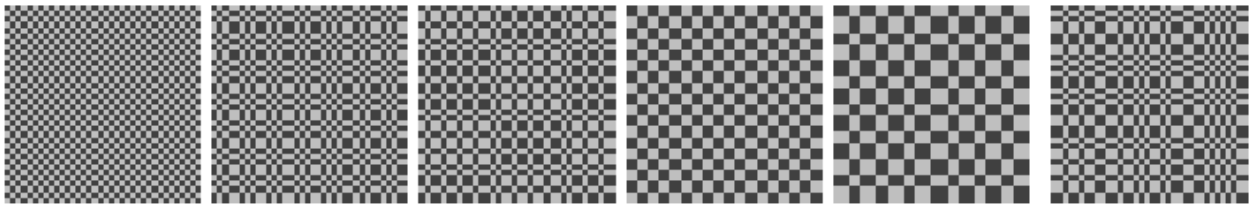

Figure 11 Left to right: A 36×36 square grid; disjoint pseudorandom FMP regions with $\alpha \in \{\sqrt[3]{2}, \sqrt{2}, 2, \sqrt{5}\}$; and disjoint random FMP regions for $\alpha = \sqrt{2}$. For $\alpha \in (1,2)$ the rectangles have sides of length 1 or 2. For $\alpha \in (2,3)$ the rectangles have sides of length 2 or 3. [Please zoom in if the images appear blurred.]

To generate the random/pseudorandom pooling regions, we need to generate two sequences first, $\{\boldsymbol{a_m}\}$ and $\{\boldsymbol{b_m}\}$. So, we can generate these two random increasing sequences by increaseing the $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$ with floor(α) or ceiling(a). For instance, for α = 1.5, the increments between $\boldsymbol{a_i}$ and $\boldsymbol{a_{i+1}}$ is 1

or 2. Same with the sequence $\{\boldsymbol{b_m}\}$. Then the pooling the ith regions can produced by $\{\boldsymbol{a_{i-1}}, \boldsymbol{a_i}\} \times \{\boldsymbol{b_{i-1}}, \boldsymbol{b_i}\}$. So, based on this, the sizes of the pooling region of FMP with α = 1.5 are 1×1, 1×2, 2×1 and 2×2, which four regions that can be seen in the figure above. For these four different size pooling regions, the max-pooling method is applied.

There are two types of FMP, random and pseudorandom. From the author, the swquence is called the pseudorandom if they are generated by the equation (18). From the figure above we can also see that the pseudorandom grids have more stable patterns.

$$\boldsymbol{a_i} = \textbf{ceiling}(\alpha(\textbf{i} + \textbf{u})), \ \alpha \in (1, 2), \text{ with some } \textbf{u} \in (0, 1) \qquad (18)$$

Implementation: For more detailed show the effects of the FMP, the author applies disjoint random FMP recursively on a color image, as shown in Figure 12.

Figure 12: Top left, ‘Kodak True Color’ parrots at a resolution of 384 × 256. The other five images are one-eighth of the resolution as a result of 6 layers of average pooling using disjoint random FMP-pooling regions α = sqrt(2)[2].

From the Figure 11, we can see that the 5 parrot images after the FMP has different level and kind of distortions. More specifically, the second image squeezed the image vertically, and the fifth image has a larger scale of the green parrot. This differences show the randmoness of the FMP.

For CIFAR-10, the paper build a more complicated network. The network architecture is stated as below in equation (19). The equation means there are 12 2-D convolutional layers and 12 FMP layers. For nth convolutional layers the number of the filters are 160n, and for each FMP layer the $\alpha = \sqrt[3]{2}$. The author obtained 3.47% error rate with 100 tests by this architecture.

$$(160nC2 - FMP\sqrt[3]{2})*12 - C2 - C1 - output \quad (19)$$

Analysis: The previous state-of-the-art architecture of the Kaggle competition is shown as equation (20). The pooling layers are using the max pooling method with window size of 2 and it reaches a test error of 4.47%.

$$(300nC2 - 300nC2 - MP2)*5 - C2 - C1 - output \quad (20)$$

For the number of filters, we can see that the model of (20) contains 9000 filters. And for the model of (19), it contains 12480 filters. So, the computational complexity of FMP architecture is higher than the Kaggle architecture. Also, the same data augmentation and dropout is used. So, the author get the conclusion that FMP can perform better than MP in CIFAR-10 dataset. Another thing needs to mention is that, with too much data augmentation or dropout, the model of FMP is easily to get under-fitting the training data.

In addition, from the paper, in many datasets like MNIST, CIFAR-100 and Chinese handwritten characters, the FMP outperforms other state-of-the-art algorithms in test accurancy.

Reasons for Better Performance: Since using the Max-pooling with window size of 2 will have some problems as stated above. Also, the model is tend to be over-fitted if the model was trained too many epochs or the dropout is not sufficient. As my point of view, this technique introduces randomness into the network, which introduces the elastic distortion of the image while doing the pooling process. As we can see in Figure 11, from the same image, we can generate down-sampled images varies with different kind of distortions and different scales. At some levels, these distortions and translations can be regarded as a kind of data augmentation since they generates different perspects of same image.

Firstly, the CNN the paper uses a large network, which contains 12480 filters. The performance of the CNN can be achieved by the proper deep network since deep neural network works better than shallow neural networks. Based on the deep network, the FMP introduces augmentation-like operations, which can improve the generalization ability of the network. The first reason makes sure the network has the ability to classify the images. And the second reason makes sure that model will not over-fit the data. So, the performance can reach the state-of-the-art.

1.3.2 Comparison with LeNet-5

For LeNet-5, from the discussion above, we can see that the accurancy

of LeNet-5 is quite low. The highest test accurancy is around 65%. We can see that LeNet-5 is easily to be over-fitting, too. This is due to there are no data augmentation or dropout is applied. So, the generalization ability of LeNet-5 is poor. Also, since LeNet-5 is designed to classify the MNIST dataset whose images are one channel grayscale handwritten digits, but not the CIFAR-10 dataset whose images are 3 channel colorful objects. So, the only two convolutional layer might not be sufficient to extract the features clearly and classify the images correctly. However, LeNet-5 has some advantages at some level. For instance, the training time is much less than other methods since the simple network architecture and less trainable parameters.

For CNN with FMP, the advantages are obvious, the test accurancy of CIFAR-10 is much higher than LeNet-5. According to the paper, the test accurancy can reach 96.43% with 100 tests applied. As to the network complexity, the paper does not give a exact number of trainable parameters of the network. However, we can know from the network it uses for CIFAR-100, with input layer size of 94 × 94 as expressed in equation (21). According to [2], the number of the weight parameters is 12 million, which is much more than LeNet-5 and needs more time to train (18 hours on GTX 780 for 250 repetition). For generalization ability, the network of CNN with FMP has a better performance on other data images. As stated above, the generalization ability is improved by the augmentation-like operation in FMP.

$$(64nC2 - FMP\ \sqrt[3]{2})*12 - C2 - C1 - output \quad (21)$$

To show the pros and cons of these two methods, the properties of them are listed in Table 1.

Table 1. Comparison between LeNet-5 and Fractional Max Pooling

Comparison	Pros	Cons
LeNet-5	-Model is simple to construct -Low computational complexity -Can reach Mediate Accuracy (around 65%) with small cost	-Not accurate enough -Low generalization ability -Not suitable for colorful images -Easy to overfit -No dropout or augmentation
Fractional Max Pooling	-Robust on preventing overfitting -High test accuracy -High generalization ability -Long time to train	-High computational complexity -Long time to train -Larger network architecture

References

[1] LeCun,Yann,etal. "Gradientbasedlearningappliedtodocumentrecognition." ProceedingsoftheIEEE86.11(1998):2278-2324

[2] Graham, B.: Fractional max-pooling. arXiv:1412.6071 (2015)