EE569 Digital Image Processing

HOMEWORK #5

Zheng Wen
zwen1423@usc.edu

Problem 1: CNN Training on LeNet-5

(a) CNN Architecture

1.CNN Components:

1) Fully connected layer: Fully connected layer is constructed by several layers of perceptron with activation function, known as multilayer perceptron (MLP). Each perceptron is fully connected with the perceptron in the next layer with different weight, and the input value of each perceptron is decided by the value of former layer, the value of weight connecting them together and the bias of each perceptron. According to the universal approximation theorem, multilayer perceptron with non-linear activation function could approximate any engineering function, so fully connected layer could be regarded as classifier in CNN, calculating non-linear combination for the features extracted by convolution layers, and fit them into different classes. The function of fully connected layer is shown below, where $a^{(i)}$ denote the activated value of ith layer, h($\cdot$) denote the activation function, W_i denote the weight matrix connecting (i-1)th layer and ith layer, b_i denote the bias used in ith layer.

$$a^{(i)} = h(W_i a^{(i-1)} + b_i)$$

2) Convolution layer: Convolution layer is used to do feature extraction in CNN. Each convolution kernel could be viewed as a filter extracting certain feature in a certain layer. Because of there are several trainable filters with different initial value in each convolution layer, so they could extract different features. The size of kernel nowadays is 3*3 typically, because of the down-sampling and the increasing in depth of the network, even 3*3 kernel could finally see a large part in the whole image in the deeper layer, so the feature could be thoroughly extracted. The operation performed in convolutional layer is the same as filtering, and a feature map stacking all the output of each kernel is obtained.

Essentially, convolution layer is the same as fully connected layer, they are all performing matrix multiple and activation, the only difference is that the convolution layer is not fully connected, and the weights in convolution layers are shared by different perceptron in the same channel. If convolution layer is converted into fully connected layer, there will be a lot of zeros in the weight matrix. Another reason using convolution layer is that convolution layer does not require the size of input, for any size of input pictures, convolution layer could serve as a slide window to extract features of the picture, which also handles the situation when training pictures have different size with test pictures, while the size of input of fully connect layer is fixed.

3) Max pooling layer: Max pooling layer is used to down-sample the feature map obtained by convolution layer, after max pooling, the max value of a certain region (typically 2*2) of the feature map is remained while others are discarded. The main purpose of pooling is to reduce the dimension of feature map and thus reduce the number of trainable parameters, computational complexity and the size of network. The reason that max value is retained is that this operation tends to capture the place which gives the strongest response, max pooling tends to retain the texture information of an image, which is useful when

doing feature extraction and also incur non-linearity. Max-pooling could also help to realize invariance, because even if there exist a small distortion, like translation, scaling or rotation, the location of each pixel is slightly changed, max pooling layer could still capture the same value in most situation. Max pooling also helps to enlarge the receptive field. Because the image is shrunk, while the size of filters in the deeper layer remains the same, so the filters in the deeper layers could see a larger region of the whole image.

4) Activation functions: Activation function is a key part in CNN, because if there is no activation function between layers, the whole CNN could be regarded as linear combination of the former layers no matter how the structure of the CNN is constructed. The existence of activation function is to incur non-linearity into CNN to help the classifier to classify the dataset, which is commonly used in other classifiers as well, like kernel functions in SVM. With the varies of input value and activation function, MLP could approach almost every engineering function, which is known as universal approximation theorem. Also, activation function makes the depth of CNN make sense. There are several activation functions, the most common one is ReLU function.

$$h(x) = max(0, x)$$

This function will let the positive value pass while kill the negative value. Similar to our brain, when two totally inverse impulse occur at the same time, CNN could only handle one at the same time, or it will be confused. ReLU function could have such a function to CNN, so it could help CNN to get a higher classification accuracy.

5) Softmax function: Softmax function is used as the activation function in the last layer of classification CNN, because the purpose of this kind of CNN is to classify different picture, the loss function of such model is typically cross entropy, which measures the difference between two probability distribution, so the output of the last layer of CNN should be normalized with the same form of probability. Softmax function will convert a vector with random value to value between 0 and 1, and the sum of the activated vector should be 1, which makes it convenient for us to compute cross entropy between the true label and the predicted score. Another reason using softmax function is that the computation of softmax function in corporate with cross entropy is convenient, the only thing to do for back propagation is to calculate the difference between true label and predicted score. The expression of softmax function is shown below,

$$S_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

where S_i denotes the ith output of a vector after going through softmax function, x_i denote the ith value in the original vector. Softmax function also gives soft decision scores for each output vector, the value of each x_i could be viewed as the probability that the input belongs to a certain class.

2. Over-fitting is a common problem in model learning, because a model is trained on a certain training set, the trainable parameters are trained by the training data and training label under the assumption that all test data share the same distribution as the training data, so the model tend to totally fit the whole training set, achieving accuracy as high as possible. But in the real problem, the test set does not necessarily have the same distribution with training set, or the training set could not represent the distribution of samples thoroughly. So, if the model totally fit the training set, which means that the accuracy on training set achieve a relatively high value, the performance on test set may be poor, which

means the generalization of model is not so good. The most common performance of over-fitting is the accuracy on training set is good while the accuracy on test set is much lower than that.

There are several ways used in CNN to avoid over-fitting. One method is called weight decay, in which a normalization term is added to the end of cost function. The normalization term is typically the sum of absolute value, called L1 regularizer, or square value of weights, called L2 regularizer. Because in an over-fitting system, the values of weight are large, so adding the normalization term in cost function could make the value of weight smaller to some degree, and thus the over-fitting is mitigated. Another method called data augmentation could also help to reduce over-fitting. One reason of over-fitting is that the number of training sample is small while the number of trainable parameters is large, if there are only little training data, a few parameters could totally fit the training set, the model certainly does not have a strong generalization because the training data is too little and could not represent the whole distribution of the dataset. So, data augmentation, such as rotation, translation and scale transform, could be apply to the given training set to increase the number of training data. Even through these image looks similar by we human, but they are totally different pictures for the computer, and they have the same distribution with the original training data, so the over-fitting is reduced. Early stop could also help to reduce over-fitting. Early stop is to calculate the accuracy on training set and validation set after each epoch, and if the error rate on validation set is about to increase as the training goes on, while the error rate on training set is still decrease, the training process should be stopped. Because under this situation, the accuracy on training set is still growing up, but the generalization of the model is decreased. So early stop is a strategy that stop training before the model becomes over-fit. Dropout is another method to overcome over-fitting, which means that the weight of each perceptron is set to 0 randomly during training. This could make the model be regarded as a combination of several different models, increase the diversity of model, and thus the generalization of model is increased, over-fitting is reduced.

3. CNNs work much better is because CNN is an end-to-end design, only the architecture of the network and several hyperparameters need to be designed by human, all jobs such as feature extraction, is finished automatically, the weight of each kernel is learned by algorithm, and thus the features are designed by backpropagation. In this way, the underlying features selected by CNNs are the most useful features for classification. While in traditional computer vision domain, the features are designed and selected by experts, people should spend a lot of time on designing features, selecting the most important ones for a given tasks. As the size of training set and the number of training labels increase, it become a tremendous job for experts to complete, and the most useful features for classification is typically unknown by human when there are a lot of classes in a huge training set. Another reason that CNNs work better in computer vision problems is the flexibility, which means that CNNs could be used for several different tasks by retraining by different dataset. However, in traditional computer vision problem, the features for different jobs are typically different. For example, in face detection, there are several Haar like features for different part of a face, but in CNN, only given the label is enough and the features are extracted by CNN automatically. And if we want to use the same CNN to detect other things such as the whole people, Haar like features will not work, but CNN still work if they are re-trained by the corresponding training set.

4. Loss function is typically multiclass cross entropy, which measures the difference between two probability distribution, is shown below:

$$C = -\sum_{i=1}^{M} y_i ln a_i$$

Where y_i is the true label of ith class, a_i is the ith activation value of the last layer of fully connected layer after softmax function. If $a_i = y_i$ for each i, the total loss of the corresponding sample is 0, and the last layer of fully connected layer outputs the right label. The less the difference between y_i and a_i, the less the value of cross entropy is. So, minimize cost function like multiclass cross entropy could make the distribution between predicted label and true label closer. In practice, the classification problem typically uses one-hot label, which means there is only one 1 in true label **y**, and the other value of **y** is 0. In this case, the cost function C could be simplified to

$$C = -ln a_j$$

where only the jth label in y is 1, the other value of **y** is 0. In backpropagation, the gradient is transmitted by the partial derivative between cost function C and the vector before activation of each layer $\boldsymbol{s}^{(i)}$ recursively, defined as $\boldsymbol{\delta}^{(i)}$:

$$\boldsymbol{\delta}^{(i)} = \frac{\partial C}{\partial \boldsymbol{s}^{(i)}}$$

The relationship between $\boldsymbol{s}^{(i)}$ and $\boldsymbol{a}^{(i)}$ is $\boldsymbol{a}^{(i)} = h(\boldsymbol{s}^{(i)})$, where h(·) denote the activation function.

In the last layer, if the activation function is softmax and the cost function is multiclass cross entropy, which is commonly used in classification problem, the gradient of the last layer is

$$\boldsymbol{\delta}^{(L)} = \frac{\partial C}{\partial \boldsymbol{s}^{(L)}} = \boldsymbol{a}^{(L)} - \boldsymbol{y}$$

For each hidden layer, the gradient is calculated by chain rule and transmitted from the layer behind it, for the jth value in vector $\boldsymbol{\delta}^{(i)}$, $\delta_j^{(i)}$, it could be computed by the following formula,

$$\delta_j^{(i)} = \frac{\partial C}{\partial s_j^{(i)}} = \sum_m \frac{\partial C}{\partial s_m^{(i+1)}} \sum_n \frac{\partial s_m^{(i+1)}}{\partial a_n^{(i)}} \frac{\partial a_n^{(i)}}{\partial s_j^{(i)}} = \sum_m \delta_m^{(i+1)} w_{mj}^{(i+1)} \dot{a}_j^{(i)}$$

Using matrix form, the formula could be rewrite as

$$\boldsymbol{\delta}^{(i)} = \left[\left(\boldsymbol{W}^{(i+1)} \right)^T \boldsymbol{\delta}^{(i+1)} \right] \odot \dot{\boldsymbol{a}}^{(i)}$$

Where $\boldsymbol{W}^{(i+1)}$ is the weight matrix in the (i+1)th layer, T denote transpose of matrix, $\dot{\boldsymbol{a}}^{(i)}$ is the derivative of the activated function in ith layer, $\odot$ denote Hadamard product. From the concept of gradient descent, to calculate the renewed value of $\boldsymbol{W}^{(i)}$ and $\boldsymbol{b}^{(i)}$, where $\boldsymbol{b}^{(i)}$ is the bias vector, the partial derivative of C verses$\boldsymbol{W}^{(i)}$ and $\boldsymbol{b}^{(i)}$ should be calculated.

$$\frac{\partial C}{\partial w_{jk}^{(i)}} = \sum_m \frac{\partial C}{\partial s_m^{(i)}} \frac{\partial s_m^{(i)}}{\partial w_{jk}^{(i)}} = \frac{\partial C}{\partial s_j^{(i)}} \frac{\partial s_j^{(i)}}{\partial w_{jk}^{(i)}} = \delta_j^{(i)} a_k^{(i)}$$

$$\frac{\partial C}{\partial b_j^{(i)}} = \sum_m \frac{\partial C}{\partial s_m^{(i)}} \frac{\partial s_m^{(i)}}{\partial b_j^{(i)}} = \frac{\partial C}{\partial s_j^{(i)}} = \delta_j^{(i)}$$

So, the weight matrix **W** and bias vector **b** for ith layer is renewed as follow:

$$\boldsymbol{W}^{(i)} = \boldsymbol{W}^{(i)} - \eta \boldsymbol{\delta}^{(i)} [\boldsymbol{a}^{(i-1)}]^T$$

$$\boldsymbol{b}^{(i)} = \boldsymbol{b}^{(i)} - \eta \boldsymbol{\delta}^{(i)}$$

Where η denote learning rate. Above process shows the backpropagation of fully connected layer, the basic principle behind backpropagation of convolutional layer is similar with it. The result is shown below:

$$\boldsymbol{\delta}^{(i)} = \left[\boldsymbol{W}_{flip}^{(i+1)} * \boldsymbol{\delta}^{(i+1)}\right] \odot \dot{\boldsymbol{a}}^{(i)}$$

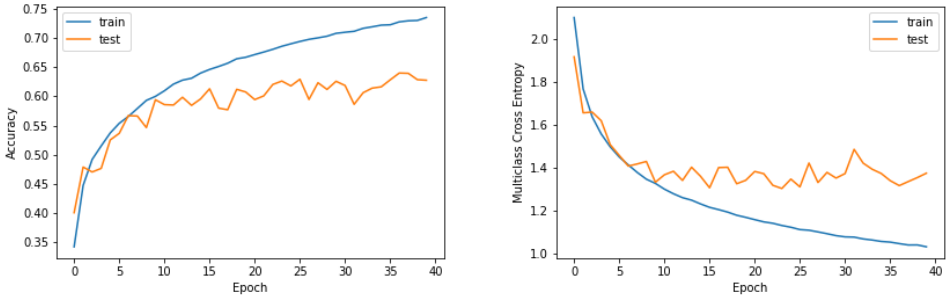
$$\boldsymbol{W}^{(i)} = \boldsymbol{W}^{(i)} - \eta \boldsymbol{\delta}^{(i)} * [\boldsymbol{a}^{(i-1)}]^T$$

$$\boldsymbol{b}^{(i)} = \boldsymbol{b}^{(i)} - \eta \boldsymbol{\delta}^{(i)}$$

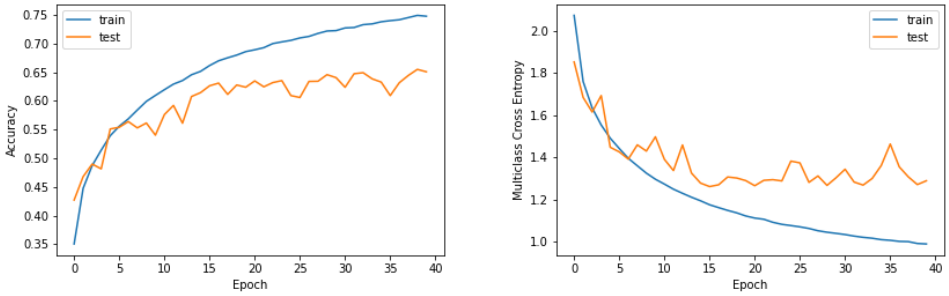
Where $\boldsymbol{W}_{flip}^{(i+1)}$ denote the flipped weight matrix, both horizontally and vertically, * denote convolution.

Backpropagation of max pooling layer is just transmitting the gradient through the corresponding position in the pooling position, leaving the gradient of the other place unchanged.

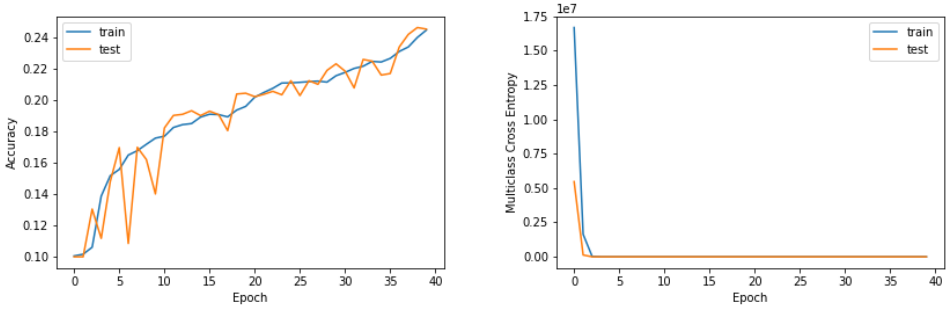
(b) CIFAR-10 Classification



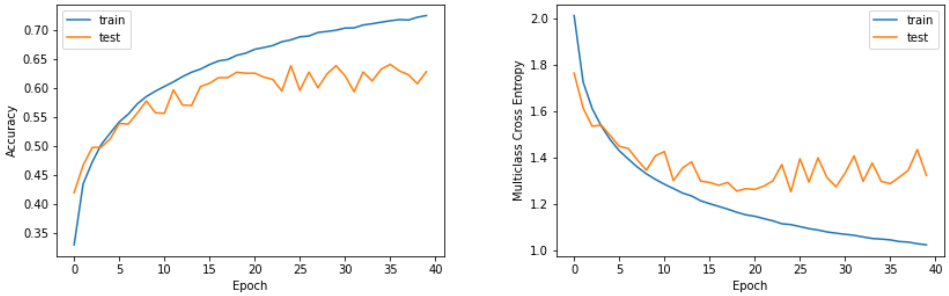
(a) 0.001 L2 regularizer, He initializer, Batch size = 128, Learning rate = 0.0005,
Train accuracy = 75.00%, test accuracy = 61.73%



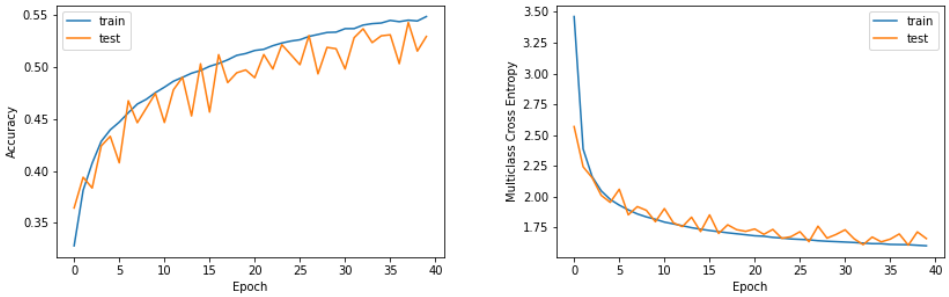
(b) 0.001 L2 regularizer, He initializer, Batch size = 128, Learning rate = 0.05
Train accuracy = 74.78%, test accuracy = 65.09%



(c) 0.001 L2 regularizer, Ones initializer, Batch size = 128, Learning rate = 0.05
Train accuracy = 24.50%, test accuracy = 24.56%



(d) 0.001 L2 regularizer, Glorot initializer, Batch size = 128, Learning rate = 0.05
Train accuracy = 72.51 %, test accuracy = 62.82 %



(e) 0.001 L1 regularizer, He initializer, Batch size = 128, Learning rate = 0.05
Train accuracy = 54.86 %, test accuracy = 52.94 %
Fig. 1 Training Curve for LeNet-5 with Different Settings

Few settings are used to initialize the network, for each training process, 40 epochs are trained to avoid over-fitting and also the performance of the networks are flattened, the optimizer is stochastic gradient descent.

In Fig. 1(a), L2 regularizer with 0.001 as parameter and He initializer are used, the batch size is 128, the learning rate is fixed as 0.0005. In contrast with Fig 1(b), where the only difference is the learning rate is fixed to 0.05. In these two experiments, the curve of the first one with smaller learning rate, have a smoother curve, but due to the smaller learning rate, the final result is not so good as the result in Fig. 1(b).

Comparing Fig. 1(b) and Fig. 1(c), the only difference is the initialization method. The weight of Fig. 1(b) is initialized by He normal initialization, which initialize the parameters by the number of nodes.

$$w \sim N\left(0, \sqrt{\frac{2}{N_{in}}}\right)$$

While the Ones initializer initializes all elements in weight matrices to 1. From the results, we could see that the impact of initialization is tremendous. When training with Ones normalization, the loss in the beginning of training is huge, even up to 1e7, and later, the increase of training accuracy and test accuracy is slower than He normalization. Because the excellent performance of CNN is decided by different kernels in each layer, and the value of the weights is decided by initial value and the gradient decent process, if the initial values of these weights are the same, these filters may have the same value, and the loss function will be easier to stuck in a local minimum, so the performance of Ones initializer is much poor than He initializer.

Comparing Fig. 1(b) and Fig.1 (d), the only difference is the initialization method. In Fig.1(d), the initializer is Glorot initializer, the weights are also initialized by normal distribution.

$$w \sim N\left(0, \sqrt{\frac{2}{N_{in} + N_{out}}}\right)$$

Because these two methods all follow the Gaussian distribution, and the difference of the number of input parameters and the number of output parameters is not so large, for LeNet-5 is a simple CNN, the training process and performance of this two is similar.

Comparing Fig. 1(b) and Fig. 1(e), the difference lies in the weight decay method, which is also known as regularizer. In Fig. 1(b), the regularizer is L2 regularizer, the key idea is adding a penalty item to the cost function. In L2 regularizer, the sum of square value of each weight is added, in L1 regularizer, the sum of absolute value of each weight is added.

$$L2: C = C_0 + \sum \|w\|^2$$

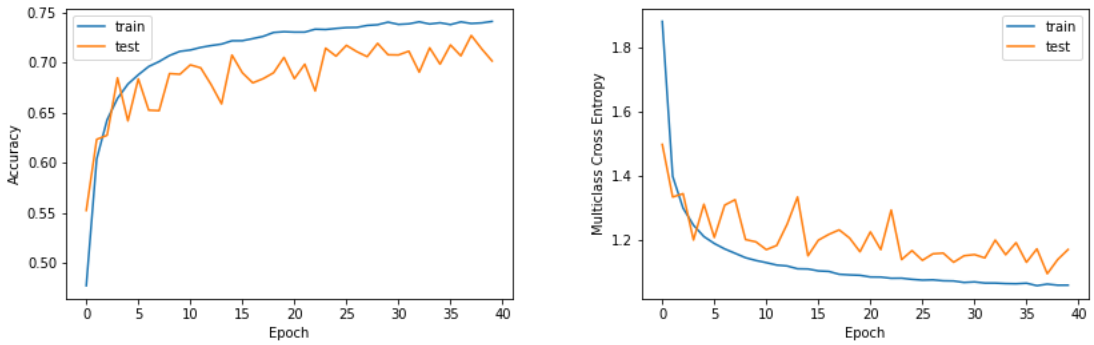
$$L1: C = C_0 + \sum \|w\|$$

Because the value of w in each kernel and fully connected layer is relatively small for the huge number of parameters, so the L1 regularizer is more severe than L2 regularizer in such a case where most of the parameters are less than 1, so, from the performance of Fig. 1(e), the convergence speed is much slower than Fig. 1(b), 40 epochs are not enough for it to fit the training set, so the performance is not as good as Fig. 1(b).

To get a better result, the base line is the parameters in Fig. 1(b), the only change is that the batch size is changed from 128 to 16, in this way, the more noise could be added into the process of stochastic gradient descent, which could help the cost function out if it get stuck in some local minima and thus get a better result. Besides, the number of filters is changed to get the full representation of the filter space. Because the filters in the first layer is 5*5*3=75, according to anchor vector principle[1], if we regard the filters as an anchor vector, and assume that all the filters are linearly independent after training, and if there are 75 filters in the first layer, the space spanned by the filters could be fully represented. This way, features in the input layer could be thoroughly extracted. So, the number of filters is set to 75 in the first layer. The

output of the first layer would become a feature map with dimension 28*28*75. Same for the second layer, because the filters in the second layer is 5*5*75 dimension, so the number of filters in the second layer is set to 1875. This network is also trained for 40 epochs.

The training curve is shown in Fig. 2, the final test accuracy is 70.19%



0.001 L2 regularizer, He initializer, Batch size = 16, Learning rate = 0.05
Train accuracy = 74.14%, test accuracy = 70.19%
Fig. 2 Training Curve of Best parameter of LeNet5

(c) State-of-the-Art CIFAR-10 Classification

Table 5: Architecture of the Large All-CNN network for CIFAR-10.

Large All-CNN for CIFAR-10

Layer name	Layer description
input	Input 126×126 RGB image
conv1	2×2 conv. 320 LeakyReLU, stride 1
conv2	2×2 conv. 320 LeakyReLU, stride 1
conv3	2×2 conv. 320 LeakyReLU, stride 2
conv4	2×2 conv. 640 LeakyReLU, stride 1, dropout 0.1
conv5	2×2 conv. 640 LeakyReLU, stride 1, dropout 0.1
conv6	2×2 conv. 640 LeakyReLU, stride 2
conv7	2×2 conv. 960 LeakyReLU, stride 1, dropout 0.2
conv8	2×2 conv. 960 LeakyReLU, stride 1, dropout 0.2
conv9	2×2 conv. 960 LeakyReLU, stride 2
conv10	2×2 conv. 1280 LeakyReLU, stride 1, dropout 0.3
conv11	2×2 conv. 1280 LeakyReLU, stride 1, dropout 0.3
conv12	2×2 conv. 1280 LeakyReLU, stride 2
conv13	2×2 conv. 1600 LeakyReLU, stride 1, dropout 0.4
conv14	2×2 conv. 1600 LeakyReLU, stride 1, dropout 0.4
conv15	2×2 conv. 1600 LeakyReLU, stride 2
conv16	2×2 conv. 1920 LeakyReLU, stride 1, dropout 0.5
conv17	1×1 conv. 1920 LeakyReLU, stride 1, dropout 0.5
softmax	10-way softmax

A new method of avoid pooling operation in CNNs is put forward in *Striving for Simplicity: The All Convolutional Net*[2], Benjamin, 2015 and achieved the second highest classification accuracy in CIFAR-10 dataset. In this paper, a new structure that contains only convolution layer is proposed. The max pooling layer is replaced by the convolution layer with stride larger than 1, and the fully connected layer is replaced by the convolution layer with size of 1*1. The structure of this network is shown in Table 5 from [2], also listed above.

The author implies that the function of pooling layer is to make the representation more invariant by the p-norm, and also suggest that the pooling layer has similar operation as convolution layer, and pooling layer could also make optimization easier by feature-wise nature. Besides, pooling layer helps to reduce the spatial dimension, making covering larger parts of the input in higher layer possible, which means pooling could make the receptive field larger. Because the property of pooling layer, the author uses a convolution layer with stride larger than 1 to be in place of pooling layer, while the original convolution layers are not changed to guarantee the effect of feature extraction. Because the deeper layer in the whole network is enough to cover a region large enough to do feature recognition, and for the purpose of unify the architecture, the fully connected layer are also replaced by 1*1 convolution layer, and the softmax function is used at the end of the whole network to produce probability representation.

The advantage of this network in comparison with LeNet-5 is its high accuracy on CIFAR-10 dataset. Using the original LeNet-5 without any modern technique and just adjust the hyperparameter of the network, LeNet-5 could not achieve a satisfying accuracy. In my experiment using LeNet-5, the test accuracy is about 70%, but this network could achieve 95% accuracy. Another advantage of this network is the uniform architecture, all the components there are convolution layers, which makes the structure of the CNN is simple in comparison with the other modern neural net, and also makes the visualization of lower layer of CNN unconditionally possible. Because in the previous CNN, the max pooling layer makes a 'switch' across the max pooling layer, and the visualized learned features must be conditioning on an input image.

The disadvantage of this network, in comparison with LeNet-5, is the high complexity, because the number of convolution layers is much larger than that of LeNet-5, and also, because the original max pooling layers are replaced by convolution layer, the number of parameters increased again, which makes the training process more costly. Besides, in the training process of the all convolution net, other layers like dropout, and some data preprocessing method is also used to achieve such a high accuracy and to overcome over-fitting, which do not exist in the training process of LeNet-5, also make the model more complex. Besides, the stride and the rate of dropout are hyperparameters which should be set by expert, costing a long time to choose and adjust them.

Reference

[1] Kuo CC. The CNN as a guided multilayer RECOS transform [lecture notes]. IEEE signal processing magazine. 2017 Apr 26;34(3):81-9.

[2] Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806. 2014 Dec 21.