**EE569 Introduction to Digital Image Processing**

**Homework Report #5**

**Name:** Boyang Xiao  **USC ID:** 3326730274  **Email:** [boyangxi@usc.edu](mailto:boyangxi@usc.edu)

# Part(a) CNN architecture

## Non-programming questions

**Q1: Describe CNN components in your own words: 1) the fully connected layer, 2) the convolutional layer, 3) the max pooling layer, 4) the activation function, and 5) the softmax function. What are the functions of these components?**

**Answer:**

1. The fully connected layer is the **CLASSIFIER** of the whole CNN. After the convolutional layers and pooling layers extract the features from the images and project the images to the feature spaces, the fully connected layers do the tensor multiplication operations and classify the original images with the extracted features.
2. The main function of convolutional layers is extracting the features from the images and project the image data from its original space to the feature spaces to do the classification.
3. The max pooling layers can reduce the size of the images and feature matrices to accelerate the computation of networks and to avoid over-fitting.
4. The activation functions can bring some non-linear operations to the CNN to improve network’s classification ability. Since the main the classifier of CNN is the fully connected layer and the operations in the fully connected layer are all linear which has limited classification ability, the activation functions can reinforce its classification ability.
5. The softmax function can mapping all the output values to the range of [0,1] and make the sum of all the values to be 1, which conforms to the probability distribution. We can choose the node with the highest value to be the final output result.

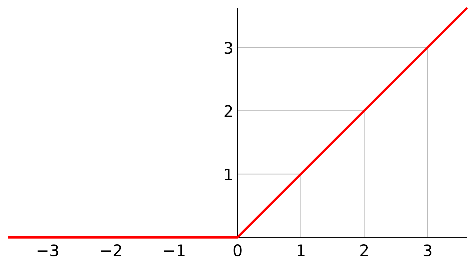
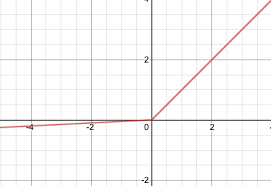
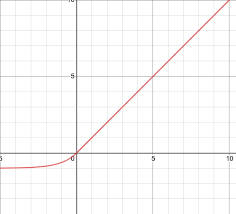
**Q2: What is the over-fitting issue in model learning? Explain any technique that has been used in CNN training to avoid the over-fitting.**

**Answer:** The over-fitting issue means the CNN model can perform very well on the training dataset but can perform worse on the testing dataset. This usually occurs when the model is too complicated and the scale of training dataset is limited. Some typical techniques to avoid over-fitting are: data augmentation, early stopping and dropout.

For example, technique of data augmentation means expanding the number of training dataset using some image modification methods such as rotation, flipping and translation. It can generate more distinguished training data using a limited scale of dataset, with which the CNN model can be better trained.

**Q3: Explain the difference among different activation functions including ReLU, LeakyReLU and ELU.**

**Answer:** The shapes of these three functions are displayed below in Figure 1.1. We can tell from the graphs that **ReLU** has zero-value before x == 0 and has a linear tilting part after x == 0. **Leaky ReLU** is much similar to ReLU but has a slightly negative linear part before x == 0. **ELU** is similar to ReLU and Leaky ReLU but has a negative non-linear part before x ==0.

(a) ReLU (b) Leaky ReLU (c) ELU

Figure 1.1 Three common activation functions

When used in the CNN model, **ReLU** can greatly accelerate the convergence process of gradient descent. **Leaky ReLU** has the same advantage as ReLU and it can also preserve the information from its negative part to avoid some neuron to be stuck/dead if it falls in the negative part of ReLU. **ELU** is a combination of ReLU and sigmoid. The right part of ELU can accelerate the gradient descent and also avoid gradient vanishing issue and the left part of ELU can preserve more information and make the CNN model more adaptive to the noise in the training dataset.

**Q4: Read official documents of different loss functions including L1Loss, MSELoss and BCELoss. List applications where those losses are used, and state why do you think they are used in those specific cases?**

**Answer:** L1Loss function and MSELoss function are commonly used in the regression situations, such as stock price prediction and temperature prediction. They are used in regression cases because they are both typical regression loss functions and are more robust and more accurate.

BCELoss function is commonly used in binary classification cases because it only has binary outputs of 0 and 1 and it makes this function more efficient.

# Compare classification performance on different datasets

## Abstract and motivation

In this part of problem, a very classical and typical Convolutional Neural Network(CNN) will be introduced and tuned, that is LeNet-5 which was one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper *Gradient-Based Learning Applied to Document Recognition.* They used this architecture for recognizing the handwritten and machine-printed characters. The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification.

In the following sections, LeNet-5 will be applied to classify some very popular and commonly-used image datasets: MNIST, fashion-MNIST and CIFAR-10, which are some earliest dataset designed for machine learning and model training. LeNet-5 model will then be tuned by trying different parameters to get higher accuracy both on training set and testing set.

## Approaches and procedures

As shown in Figure 2.1 below, LeNet-5 has a structure of 2 convolutional layers (followed by 2 max-pooling layers respectively) and three fully connected layers. The convolutional kernel window size is 5×5 and all the max-pooling window size is 2×2 (which can reduce the size to be half). The first convolutional layer has 6 filter kernels and the second convolutional layer has 16 filter kernels. The neuron numbers for the three fully connected layers are 120, 84 and 10, respectively.

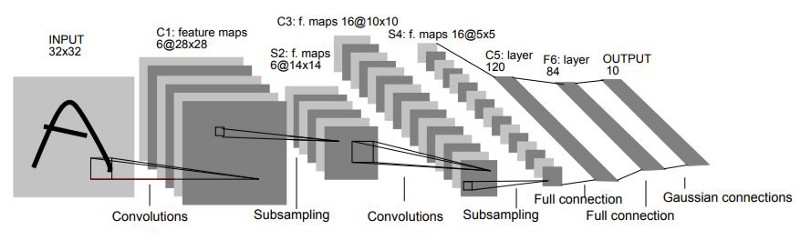


Figure 2.1 LeNet-5 network structure

The datasets MNIST, fashion-MNIST and CIFAR-10 are downloaded and separated into two groups: training set and testing set. The training set is used to train the CNN model and to adjust the weights of the neurons with its labels. The testing set does NOT involve in the training process of the model but only tests the accuracy of the model after each epoch.

The main hyper-parameters that will be tuned are the **learning-rate**, the **optimizer** and the **momentum**. Training results with different hyper-parameter settings will be shown and compared.

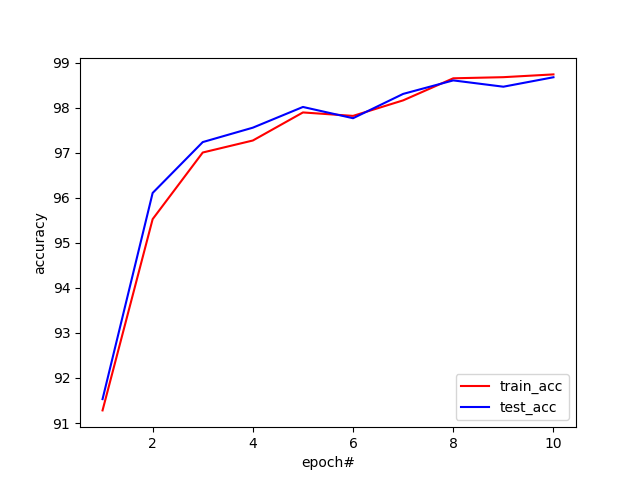
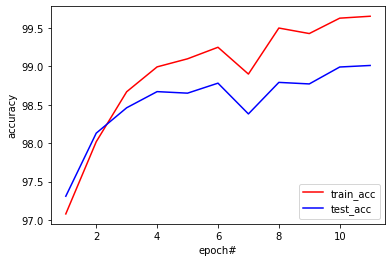
## Experiment results on dataset MNIST

The three parameter settings for experiments on MNIST are shown in Table 2.1 below. The final training accuracy and the testing accuracy are also shown in Table 2.1. The max-epoch numbers are all 10.

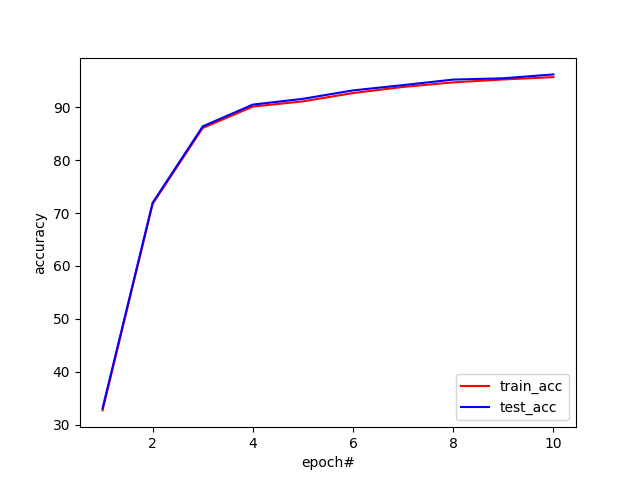
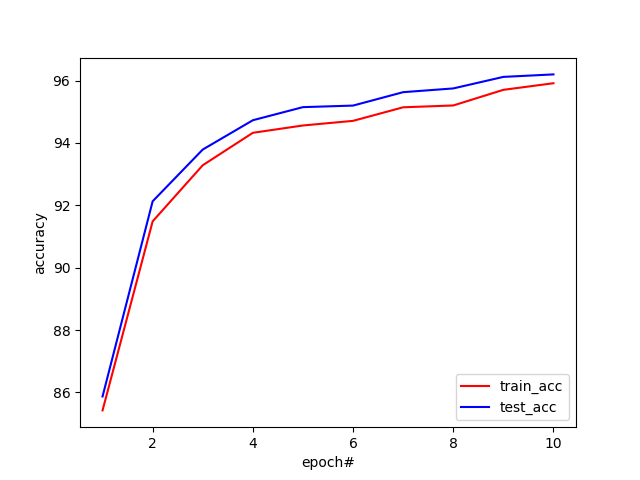
Table 2.1 Parameters settings and corresponding results for MNIST dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NO.** | **Learning rate** | **Momentum** | **Weights decay** | **Training accuracy** | **Testing accuracy** |
| **1** | 0.001 | 0.9 | 0 | 98.743 | 98.680 |
| **2** | 0.005 | 0.9 | 0 | 99.650 | 99.010 |
| **3** | 0.001 | 0.5 | 0 | 95.655 | 96.170 |
| **4** | 0.001 | 0.9 | 0.05 | 95.917 | 96.200 |

The training performance curves for the parameter settings above are shown in Figure 2.2 below, where each subplot’s x-axis represents the epoch number and the y-axis represents the accuracy for both the training set and the testing set.

(a) Setting #1 (b) Setting #2

(c) Setting #3 (d) Setting #4

Figure 2.2 Performance curves for the 4 different parameter settings

From the results above, parameter setting#1 is the most common setting with all the default parameters setting and this is the standard setting. Compared to setting#1, setting#2 uses a higher learning rate and the CNN model’s accuracy can rise much faster. However, the training process for setting#2 is not stable and has a phase of decreasing (but increases anyway after that). This is because the training rate is too high and when the accuracy reaches some level, the training is not able to get more accurate to reach the optimized point. But this high learning rate can also make the final accuracy higher and make the training process faster than the others anyway.

Setting#3 uses a lower momentum value compared to setting#1, which mainly influences the initialization of the whole CNN model. As a result, the starting accuracy (the first several epoch) is quite low because the bad choice of the momentum value, but it can still reach a higher accuracy after the training process and the testing accuracy is closer to the training accuracy, which means that the model is more robust to the unfamiliar data. However, the final accuracy is still lower than the previous settings.

Setting#4 uses a non-zero weights-decay value compared to setting#1 which uses the default zero-value decay, and this weights-decay parameter mainly influences the L2 regularization of the model. With this non-zero decay value, the training process is faster but the final accuracy is lower.

## Experiment results on dataset fashion-MNIST

The three parameter settings for experiments on fashion-MNIST are shown in Table 2.2 below. The final training accuracy and the testing accuracy are also shown in Table 2.2. The

1. Table 2.2 Parameters settings and corresponding results for fashion-MNIST dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NO.** | **Learning rate** | **Momentum** | **Weights decay** | **Training accuracy** | **Testing accuracy** |
| **1** | 0.001 | 0.9 | 0 | 98.990 | 98.650 |
| **2** | 0.005 | 0.9 | 0 |  |  |
| **3** | 0.001 | 0.5 | 0 |  |  |
| **4** | 0.001 | 0.9 | 0.05 |  |  |

