EE569 Digital Image Processing

HOMEWORK #5

Issued: 03/28/2022 Due: 11:59PM, 04/14/2022

Problem 1: CNN Training on LeNet-5 (100%)

1.1 Motivation

CNN (Convolutional Neural Networks) are a form of neural network that is one of the finest learning algorithms for interpreting visual content and excels in image segmentation, classification, detection, and retrieval tasks.

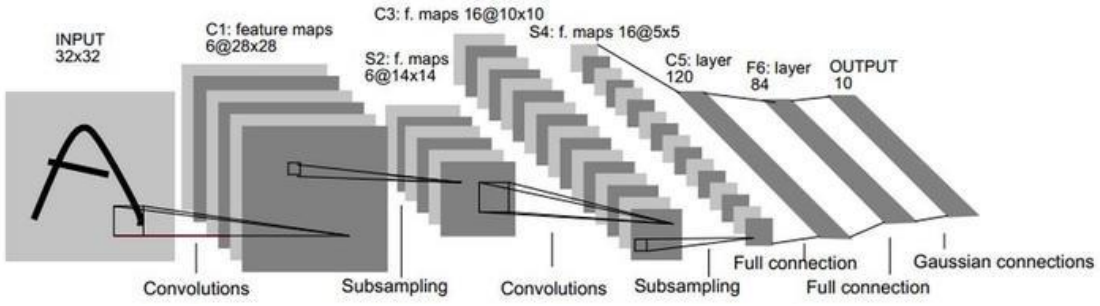
CNN is a feedforward neural network with a convolutional structure that can help the deep network use less memory. Three important actions - local perceptual field, weight sharing, and pooling layer - substantially minimize the amount of network parameters and alleviate the model's overfitting issue. Several convolutional and pooling layers are often used, and the convolutional and pooling layers are set alternately, i.e., one convolutional layer is linked to a pooling layer, followed by another convolutional layer, and so forth. CNN is primarily utilized in image recognition (CV) applications such as image classification and retrieval, target localization detection, target segmentation, face recognition, bone identification, and tracking. MNIST handwritten data recognition, cat and dog battles, ImageNet LSVRC, and other examples of bone recognition and tracking Natural language processing and speech recognition can also be used with it.

LeNet-5 is a rudimentary convolutional neural network, but it incorporates the core deep learning modules: convolutional layer, pooling layer, and complete linkage layer, which serve as the foundation for other deep learning models.

1.2 Approach

1.2.1 CNN Architecture

This is a common architecture: first, the input picture is passed through a convolution layer, and then the convolution information is processed by pooling, using the max pooling method. The information after the second processing is then sent to two fully connected neural layers through the same method, which is also a general two-layer neural network layer, and finally a classifier is added for classification prediction.



1.2.2 Compare classification performance on different datasets

Step 1. Importing the necessary packages.

Step 2. Create a function to calculate the overall data_loader prediction accuracy and plot the performance curves.

Step 3. Create a function to train model.

Step 4. Create a function to test model.

Step 5. Create a function including the train model function and test model function to define the entire training loop.

Step 6. Defining the LeNet-5 network model.

Step 7. Define hyperparameters, download MNIST/Fashion-MNIST/CIFAR-10 datasets.

Step 8. Print the loss and accuracy for each epoch and plot performance curves.

1.2.3 Analysis on confusion classes and hard samples

Step 1. Create a function to generate a normalized 10 * 10 confusion matrix on MNIST/ Fashion-MNIST/CIFAR-10 dataset.

		Predicted condition	
	Total 8 + 4 = 12	Cancer	Non-cancer
Actual condition	Cancer	6	2
	Non-cancer	1	3

Step 2. Show the top three pairs of classes.

1.2.4 Classification with noisy data

Step 1. Importing the necessary packages and set noise level.

Step 2. Create functions to build uniform noise or pair noise to randomly map a noisy label to another label.

Step 3. Create a function to implement multi-label hybrid mapping based on transformation matrix.

Step 4. Create a function to set probability P for label mapping.

Step 5. Create a function to calculate the overall data_loader prediction accuracy and plot the performance curves.

Step 6. Create a function to train model including constructing the function of label noise.

Step 7. Create a function to test model.

Step 8. Create a function including the train model function and test model function to define the entire training loop.

Step 9. Defining the LeNet-5 network model.

Step 10. Define hyperparameters, download MNIST datasets.

Step 11. Print the loss and accuracy for each epoch and plot performance curves.

Step 12. Generate the normalized confusion matrix between the true label and the noisy label.

1.3 Results

(a)

1.

1) The fully connected layer is similar to the part of a traditional neural network and is used to output the desired result;
2) The convolutional layer is a network layer used to extract local features of an image, and it achieves the extraction of local features in an image by convolutional kernel;
3) Pooling layer is used to significantly reduce the parameter magnitude (dimensionality reduction);
4) The activation function is used to determine the output of each neuron;
5) The softmax function is used in the last layer of the convolutional neural network to classify the data.

2.

Neural network overfitting refers to a neural network model that performs well on the training set but does not perform well when predicting new data. Regularization is the first strategy to try when overfitting has to be decreased. This method includes adding an extra equation to the loss function, which might complicate the model. Simply said, it uses a number in the weight matrix that is excessively large, attempting to limit its flexibility while simultaneously pushing it to design solutions based on many attributes.

3.
ReLU:
Advantages: highly nonlinear. Speeds up convergence, alleviates gradient disappearance and explosion problems, simplifies computation.

Disadvantages: Because it resets all negative inputs to 0, it can be brittle during training and can quickly lead to neuron inactivation, resulting in neurons not activating again at any data point. Because the gradient for activation in ReLU ($x<0$) is 0 at this point, the weights are not modified during descent.

LeakyReLU:
LeakyReLU is a ReLU variation with an altered response to the input less than 0 component, which mitigates ReLU's sparsity since we may tune the coefficient a to ensure a weak output when the input is less than 0.

Advantage: alleviates the problem of neuron death due to ReLU.

Cons: Having a negative output leads to a less powerful nonlinearity than ReLU, and the effect is not as good as Sigmoid in some classification tasks, not to mention ReLU.

ELU:
It, like batch regularization, can bring the data mean closer to zero. The computing complexity is lower than that of batch regularization. It has a negative component when compared to relu and is more noise resistant than Leaky-ReLU. This is due to the linear component allowing ELU to minimize gradient disappearance, while the negative component allowing ELU to be more resistant to input changes or noise.

4.
L1Loss: Generates a criteria for calculating the mean absolute error (MAE) between each element in the input x and the goal y.

MSELoss: Generates a criteria for calculating the mean squared error (square of L2 parametric) between each element in the input x and the goal y.

BCELoss: Generates a criteria for calculating the Binary Cross Entropy between the goal and input probability. Taking such qualities into account, L1Loss and MSELoss may be used to quantify the absolute distance between distinct labels, whilst BCELoss

is commonly employed to construct cross entropy and probability estimations.

(b)
1.
1)
Hyperparameters setting is shown below:
RANDOM_SEED = 42
LEARNING_RATE = 0.001
BATCH_SIZE = 32
N_EPOCHS = 15

The table of performance of this hyperparameters setting is shown below:

No.	Training accuracy	Testing accuracy
1	99.35	99.69
2	99.78	99.41
3	99.76	99.40
4	99.73	99.35
5	99.71	99.41
Best accuracy	99.78	99.69
Mean accuracy	99.67	99.45
Std accuracy	0.16	0.12

The performance curves:

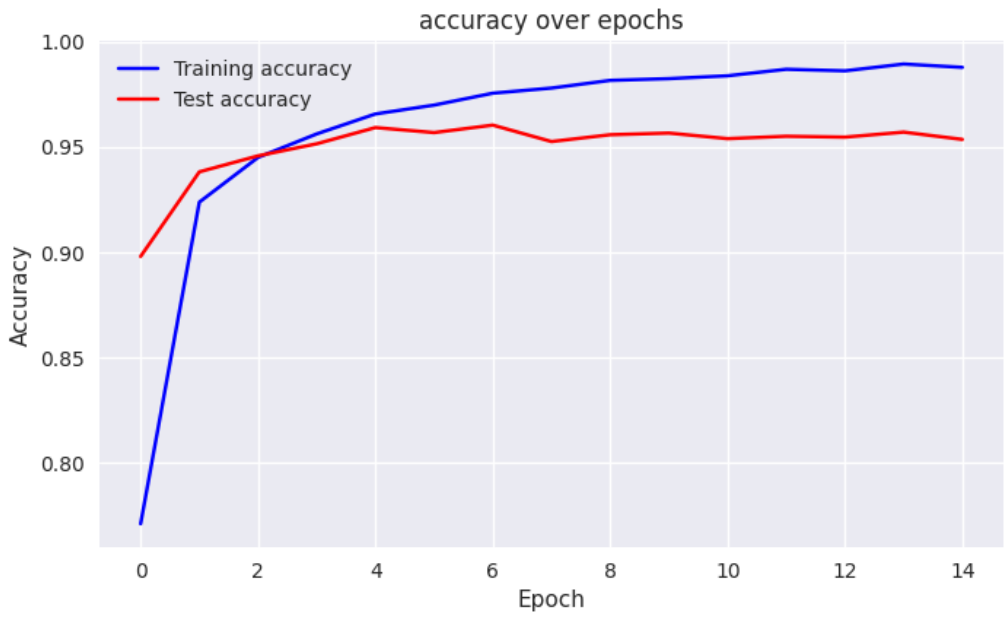


Figure 1: Performance curves under first hyper-parameter settings

2)
Hyperparameters setting is shown below:
RANDOM_SEED = 42

LEARNING_RATE = 0.005
BATCH_SIZE = 32
N_EPOCHS = 15

The table of performance of this hyperparameters setting is shown below:

No.	Training accuracy	Testing accuracy
1	97.89	98.25
2	98.04	98.39
3	97.82	98.12
4	97.94	98.30
5	97.90	98.23
Best accuracy	98.04	98.39
Mean accuracy	97.92	98.26
Std accuracy	0.07	0.09

The performance curves:

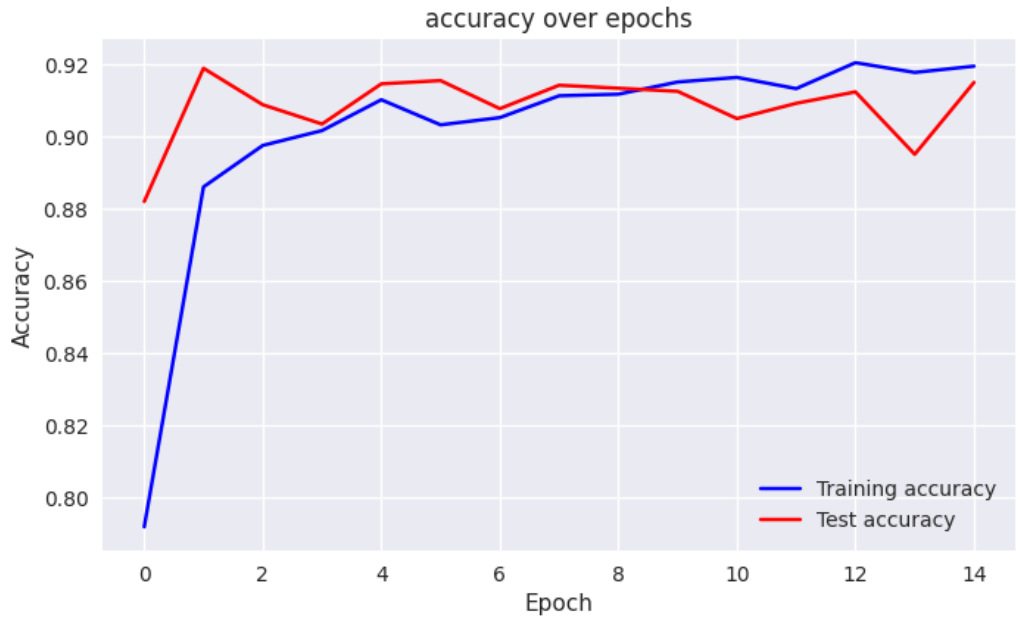


Figure 2: Performance curves under second hyper-parameter settings

3)
Hyperparameters setting is shown below:
RANDOM_SEED = 42
LEARNING_RATE = 0.001
BATCH_SIZE = 8
N_EPOCHS = 10

The table of performance of this hyperparameters setting is shown below:

No.	Training accuracy	Testing accuracy
1	99.31	99.16

2	99.36	99.21
3	99.30	99.17
4	99.20	99.06
5	99.32	99.17
Best accuracy	99.36	99.21
Mean accuracy	99.30	99.15
Std accuracy	0.05	0.05

The performance curves:

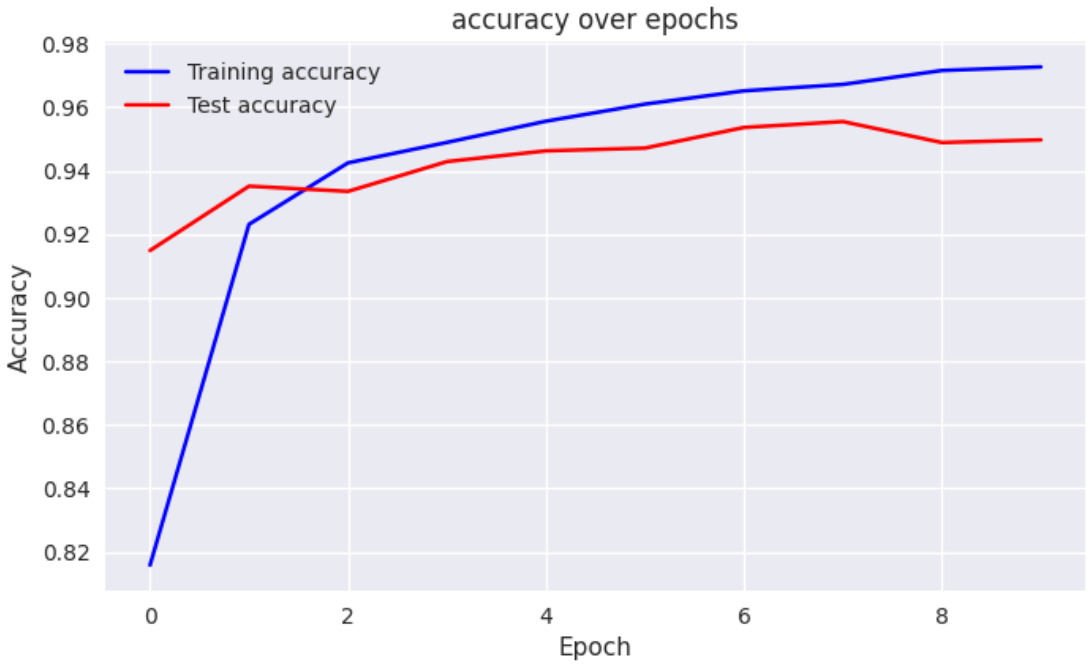


Figure 3: Performance curves under third hyper-parameter settings

With the adjustment of hyperparameters we can find that the lower the learning rate, the higher the accuracy, the larger the training epoch, the higher the accuracy, and the larger the batch size, the higher the accuracy.

2. We can find the best parameter setting is RANDOM_SEED = 42, LEARNING_RATE = 0.001, BATCH_SIZE = 32, N_EPOCHS = 15. The performance curves is shown below:

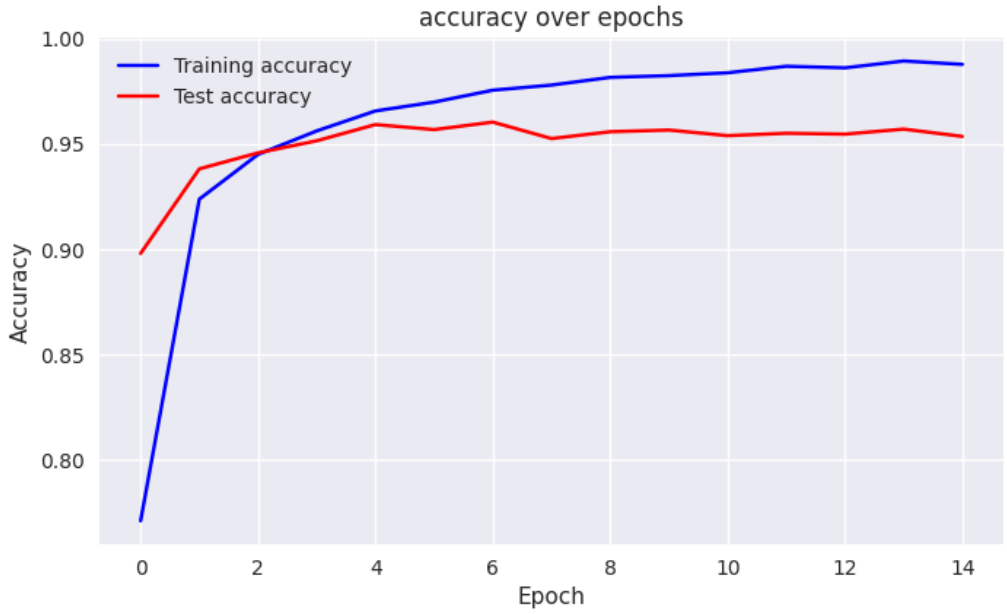


Figure 4: Performance curves under best hyper-parameter settings for MNIST

3.
Hyperparameters setting is shown below:
RANDOM_SEED = 42
LEARNING_RATE = 0.001
BATCH_SIZE = 32
N_EPOCHS = 30

The table of performance of this hyperparameters setting is shown below:

No.	Training accuracy	Testing accuracy
1	90.76	90.50
2	90.41	90.27
3	90.82	90.50
4	90.78	90.44
5	90.80	90.39
Best accuracy	90.82	90.50
Mean accuracy	90.71	90.42
Std accuracy	0.15	0.09

The performance curves:

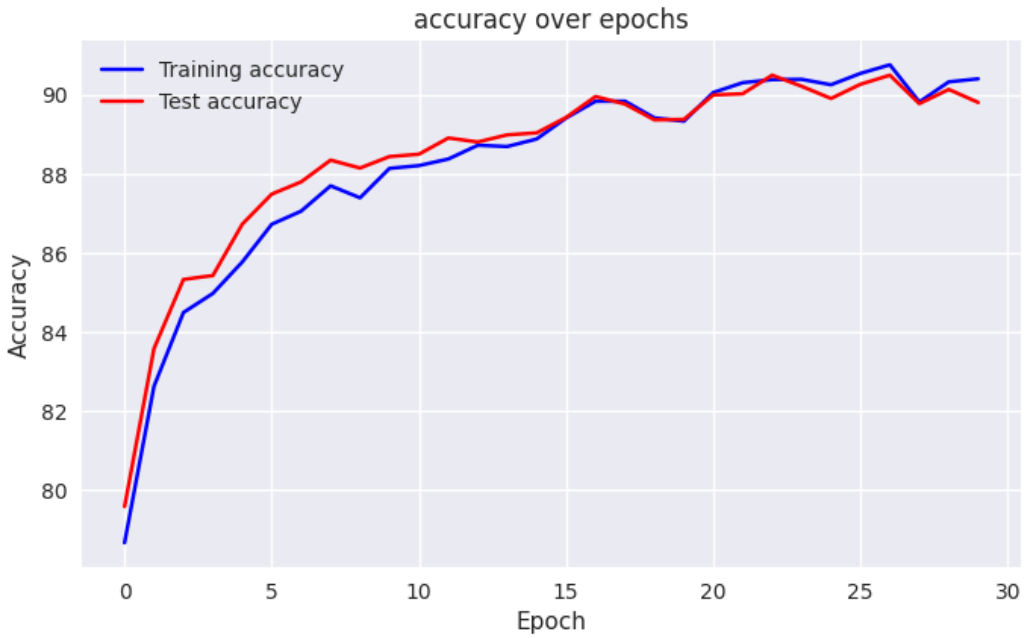


Figure 5: Performance curves under best hyper-parameter settings for Fashion-MNIST

4.

Hyperparameters setting is shown below:

RANDOM_SEED = 42

LEARNING_RATE = 0.001

BATCH_SIZE = 32

N_EPOCHS = 15

The table of performance of this hyperparameters setting is shown below:

No.	Training accuracy	Testing accuracy
1	69.98	66.60
2	69.75	66.62
3	69.98	66.60
4	69.86	66.48
5	69.88	66.59
Best accuracy	69.98	66.62
Mean accuracy	69.69	66.58
Std accuracy	0.42	0.05

The performance curves:

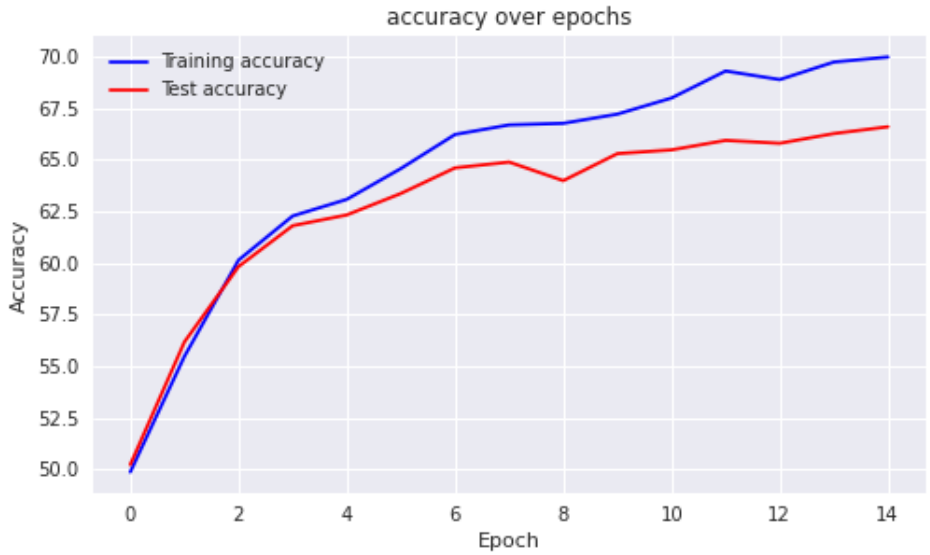


Figure 6: Performance curves under best hyper-parameter settings for CIFAR-10

5.

Based on the training result analysis of different training data we found that CIFAR-10 is harder to recognize than Fashion-MNIST and Fashion-MNIST is harder to recognize than MNIST, so there will be inconsistency in the final training accuracy.

(c) Analysis on confusion classes and hard samples

1.

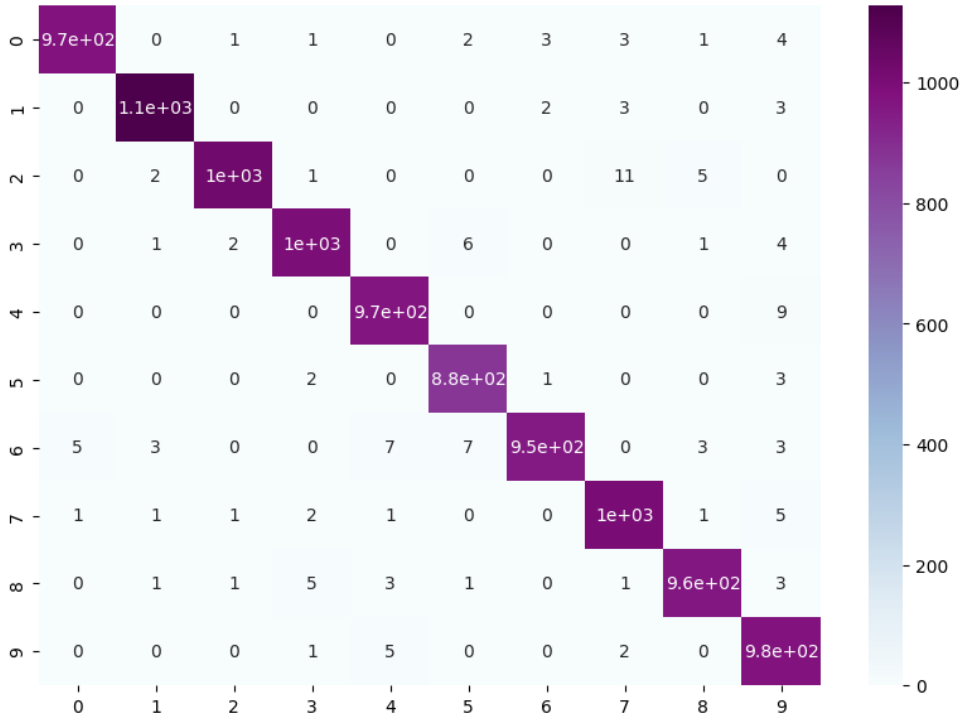


Figure 7: Normalized confusion matrix for MNIST

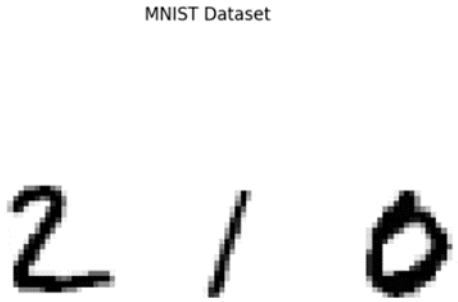


Figure 8: Top three confused pairs of classes for MNIST

It can be seen that the three types of digital images that are not easily recognized all differ greatly from the standard digital images.

2.

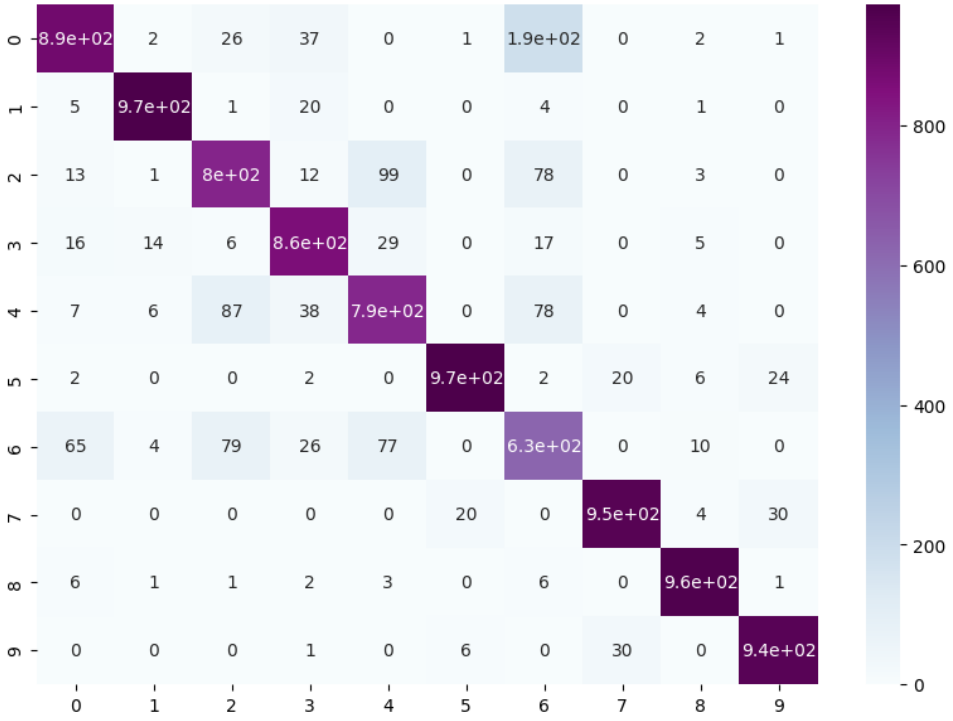


Figure 9: Normalized confusion matrix for Fashion-MNIST

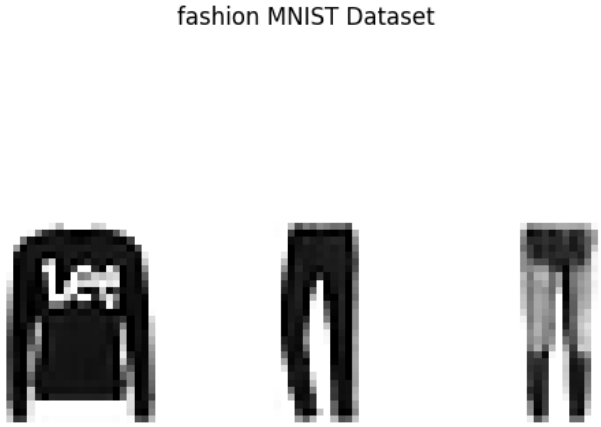


Figure 10: Top three confused pairs of classes for Fashion-MNIST

It can be seen that the three types of images that are not easily recognized are all different from the standard image and contain more noise.

3.

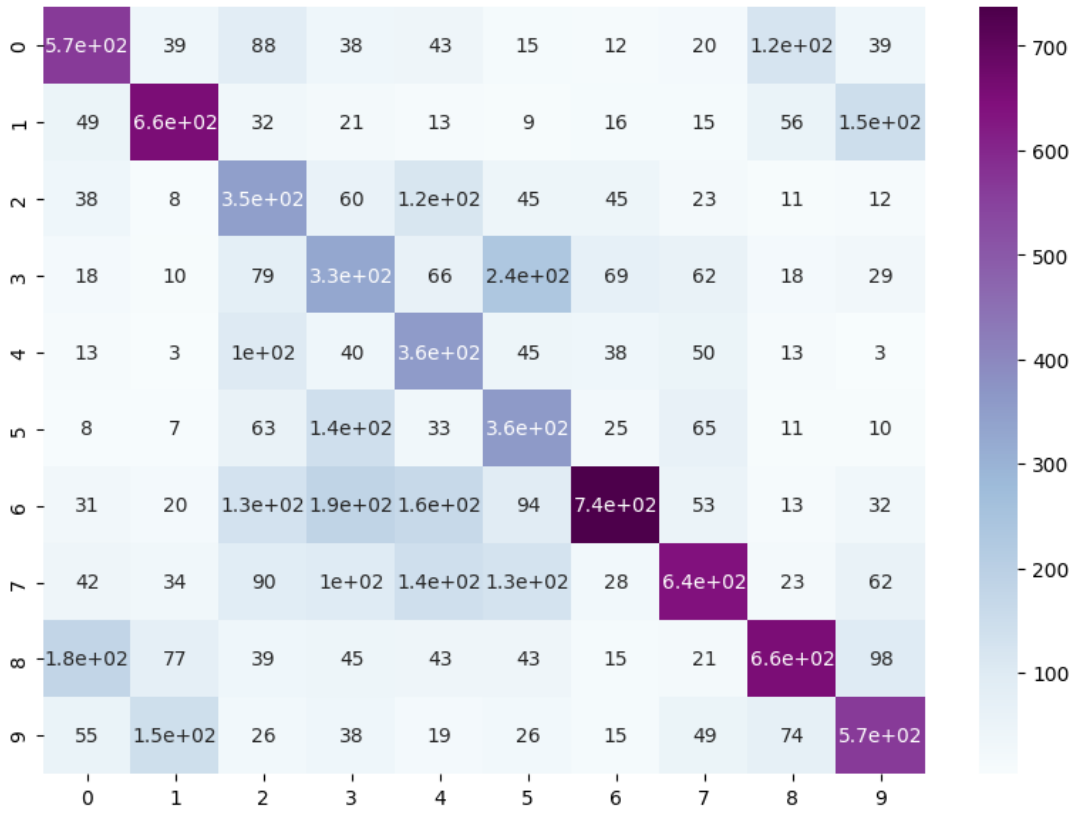


Figure 11: Normalized confusion matrix for CIFAR-10

Cifar10 Dataset

Figure 12: Top three confused pairs of classes for CIFAR-10

It can be seen that the three types of images that are not easy to identify are all different from the standard image and have serious distortion.

(d) Classification with noisy data

1.

The method is describe in 1.2 Approach, 1.2.4 above. The normalized confusion matrix for ϵ = 40% is shown below:

```
matrix([[0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0. ],
        [0.2, 0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0. ],
        [0.2, 0.2, 0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0. ],
        [0.2, 0.2, 0.2, 0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0. ],
        [0.2, 0.2, 0.2, 0.2, 0.6, 0.2, 0.2, 0.2, 0.2, 0. ],
        [0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.2, 0.2, 0.2, 0. ],
        [0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.2, 0.2, 0. ],
        [0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.2, 0. ],
        [0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0. ],
        [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.6]])
```

Figure 13: Normalized confusion matrix for ϵ = 40%

2.

ϵ = 0%

No.	Testing Accuracy
1	97.79
2	97.77
3	97.78
4	97.79
5	97.77

Mean	97.78
Standard Deviation	0.01

The performance curves:

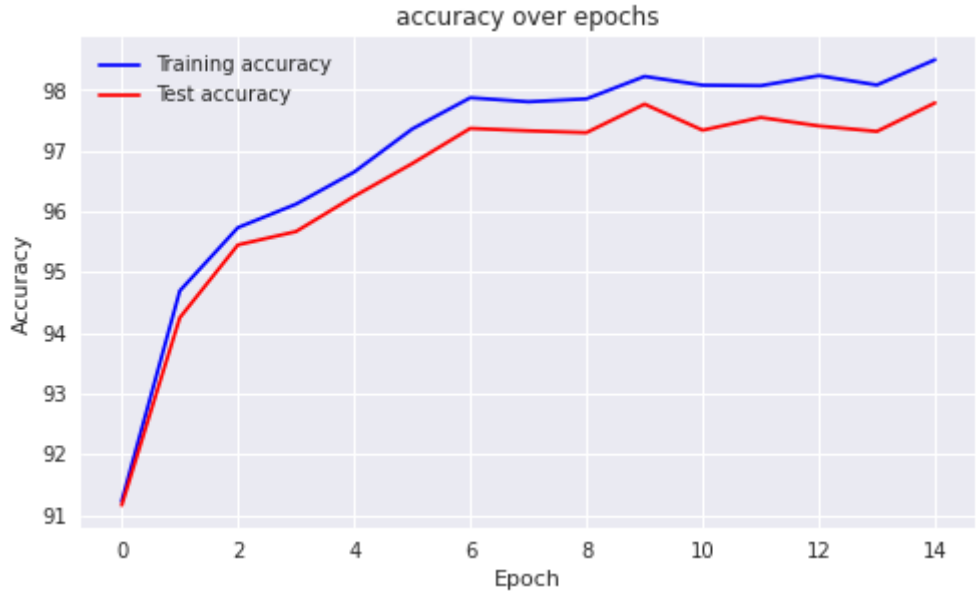


Figure 14: Performance curves with $\epsilon = 0\%$

$\epsilon = 20\%$

No.	Testing Accuracy
1	97.59
2	97.52
3	97.58
4	97.64
5	97.59
Mean	97.58
Standard Deviation	0.04

The performance curves:

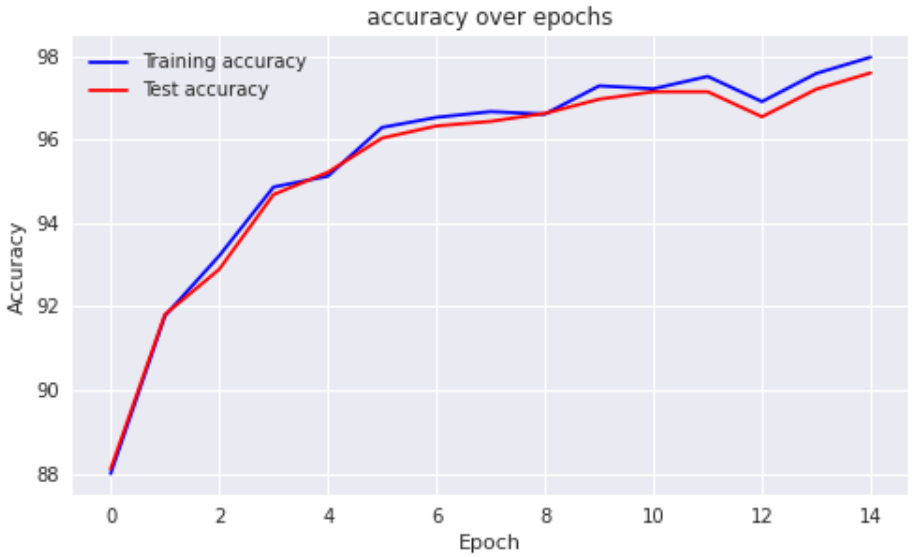


Figure 15: Performance curves with $\epsilon = 20\%$

$\epsilon = 40\%$

No.	Testing Accuracy
1	96.86
2	96.92
3	96.87
4	96.86
5	96.91
Mean	96.88
Standard Deviation	0.03

The performance curves:

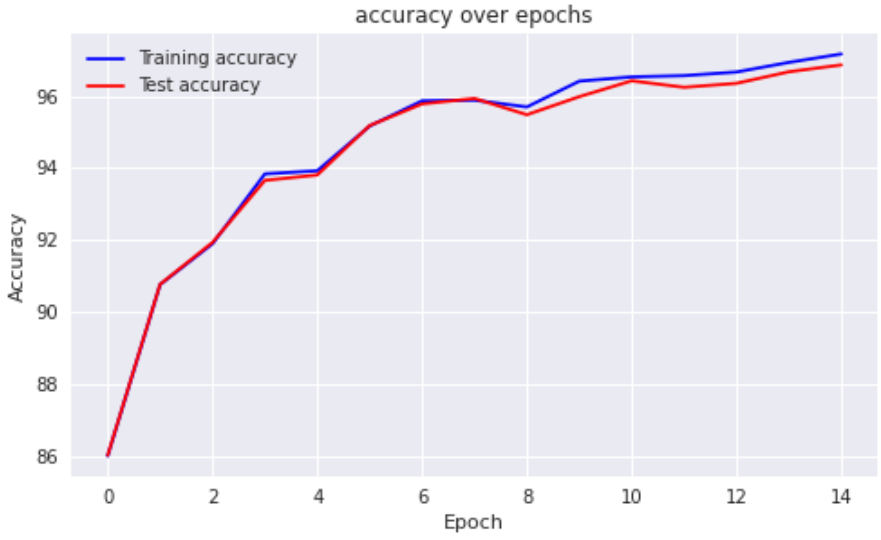


Figure 16: Performance curves with $\epsilon = 40\%$

$\epsilon = 60\%$

No.	Testing Accuracy
1	95.05
2	95.07
3	95.05
4	95.18
5	95.01
Mean	95.07
Standard Deviation	0.06

The performance curves:

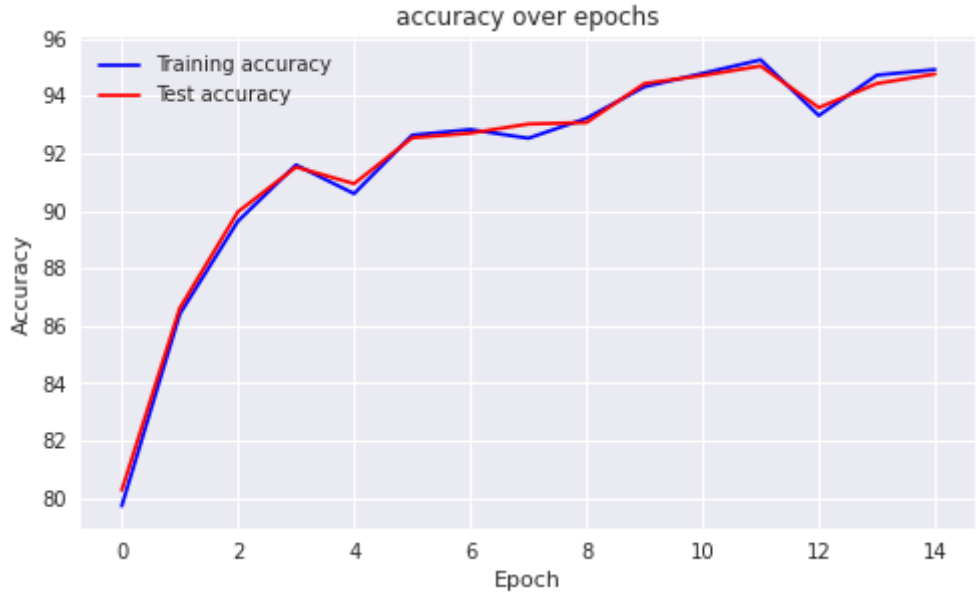


Figure 17: Performance curves with $\epsilon = 60\%$

$\epsilon = 80\%$

No.	Testing Accuracy
1	59.31
2	59.08
3	59.81
4	59.15
5	59.56
Mean	59.38
Standard Deviation	0.27

The performance curves:

Figure 18: Performance curves with ϵ = 80%

3.

It can be seen that the accuracy of the trained model gradually decreases with the injection of noise.

1.4 Discussion

Through experiments we found that the MNIST dataset identifies the data more easily than the Fashion-MNIST dataset, which in turn identifies the data more easily than the CIFAR-10 dataset.

According to the results of analyzing the confusion classes, the three data sets each have three less identifiable digital images that are different from the standard digital images.

When we train LeNet-5 with different noise level noisy label, we can find that the testing accuracy decline gradually with the injection of noise.