HW5_Neural Networks for Classifying Fashion MNIST

Yanjie Liu

Abstract:

The goal of this report is to construct a basic neural network and explore its application in recognizing different items from MNIST dataset. The basic process is to build a neural network classification model and optimize it to reach higher accuracy of classification.

Introduction and Overview

In order to classify the different items from MINIST dataset, we need to know how our brain does this classification work. There are billions of neurons which connect to each other in our brain. And each neuron receives inputs and produces appropriate outputs from other connected neurons. The method which we use in this report is to imitate the biological process and its concept is “neural network”. The neural network in this report is designed to learn from a large training dataset and through each layer, it will automatically construct the more and more complicated and abstract learning structures of images. Also, the concept of artificial intelligence is the extension and application of neural network and the applications of neuron network have already been very common in our daily life like Siri service from Apple and Alexa service from Amazon.

Theoretical Background

Linear Regression

Linear regression is one of the most important concept of neural network. The basic concept is to take some data and find a line of best fit. The model of linear regression is:

$$y = mx + b \tag{1}$$

The function to determine which parameters make the model best fit:

$$\hat{y}_j = mx_j + b \tag{2}$$

$\hat{y}_j$ is the point on the line corresponding to the data point x_j . And the most common choice for linear regression is the mean squared error:

$$\frac{1}{n}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2 \tag{3}$$

The above equation referred to as simple linear regression, since this is a linear approach to modeling the relationship between a scalar response and another explanatory variable. For the multiple linear regression, the equation is:

$$y = a_1x_1 + a_2x_2 + \dots a_nx_n + b \tag{4}$$

Logistic Regression

Logistic Regression is the most basic model for a classification problem. This concept is used for binary classification. In another word, the logistic function will better fit binary classification dataset. The logistic function is:

$$y = \frac{1}{1 + e^{-(mx+b)}} \tag{5}$$

y is also the probability which the particular data point belongs to class 1($y = 1$). And $1 - y$ is the probability which the particular data point belongs to class 0 ($y = 0$). Then the loss function is also different from linear regression:

$$\begin{cases} 1 - p_j & y_j = 0 \\ p_j & y_j = 1 \end{cases} \tag{6}$$

In this case, we will maximize the logarithm of these functions for technical reasons. Then the loss function is:

$$\begin{cases} \ln(1 - \mathrm{p_j}) & y_j = 0 \\ \ln(\mathrm{p_j}) & y_j = 1 \end{cases} \tag{7}$$

Neural Network

The concept of neural network was created by Warren McCulloch and Walter Pitts(1943). And the neural network is a complicated function which can relate all x and y variables. It is composed by input layers, hidden layers, and output perceptron layers. The hidden layers are simply the layers that are between input and output layers. And the transformation of inputs occur here. The more layers are added in hidden layers, the more complicated and more abstract level of decision the computer will make. The number of layers and width of the model are different from models to models, which will lead to extremely different behavior and output for different models.

Algorithm Implementation and Development

Set-up

Before the start of classification, we need to load the MNIST dataset into Matlab and modify the size of dataset to build neural network model. In order to do the classification, we need to separate the data into data set and label set. Then we will cross-validate the training dataset and label set and we will have 55000 training sets and 5000 test sets.

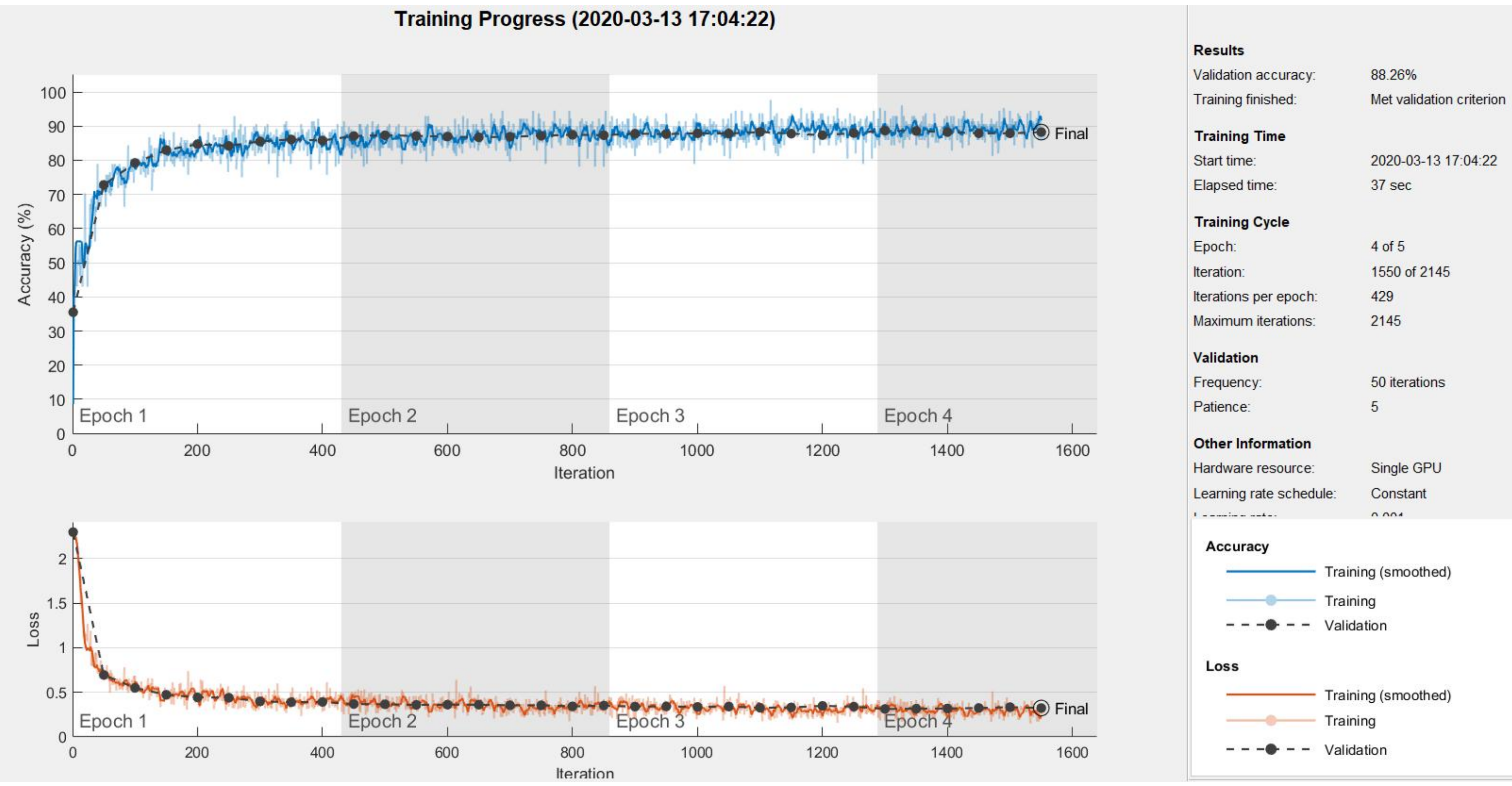
Build Neural Network Model

The main part of the neural network model is the architecture of neural networks for deep learning. To define a fully connected neural network architecture and a convolutional neural network architecture for classification, we need image input layer, fully connected layer relu layer,(convolutional 2d layer,) soft max layer, and classification layer. The next step is to optimize the train process with different hyperparameters. We need to modify the number of max epochs, initial learning rate, L2 regularization, and validation data. The last step is to use the MNIST dataset to train our model. Then repeat the previous step to change the hyperparameters and to reach higher accuracy.

Computational Result

Part 1

In this case, our model will classify 10 different items from each other. In this part, we can only use fully-connected layers to construct our neural network. For the depth and width of our neural network, we consider about the effect from the size of dataset which is 28*28 and choose 382 and 196 as the number of variables in hidden layers. The validation accuracy is 88.26% as it shown in figure 1. And the confusion matrix is figure 2. From figure 2, we know that this model behave relatively bad in recognizing item 6 and item 0.



Results
Validation accuracy: 88.26%
Training finished: Met validation criterion

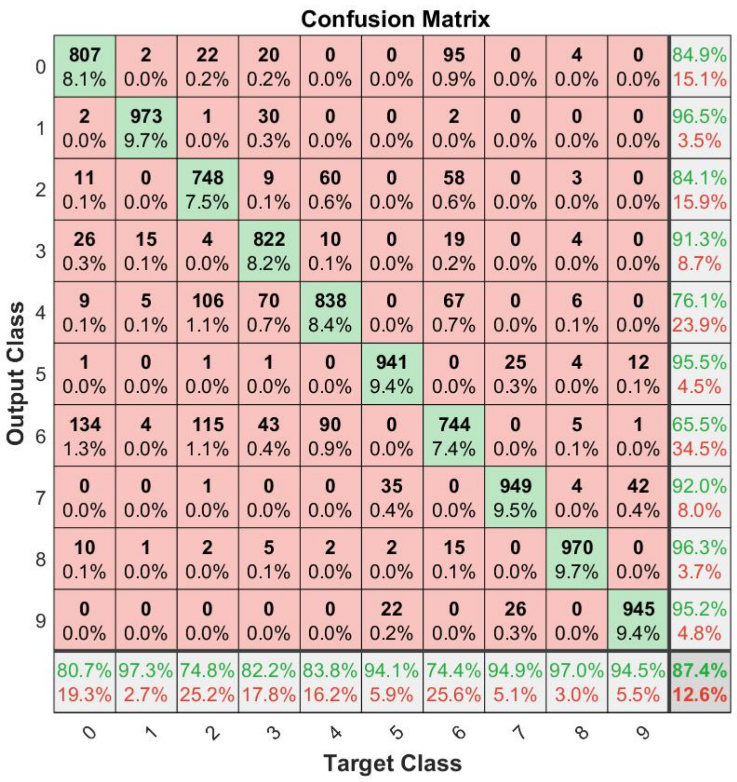
Training Time
Start time: 2020-03-13 17:04:22
Elapsed time: 37 sec

Training Cycle
Epoch: 4 of 5
Iteration: 1550 of 2145
Iterations per epoch: 429
Maximum iterations: 2145

Validation
Frequency: 50 iterations
Patience: 5

Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant

Figure 1: the training process of neural networks with only fully-connected layers

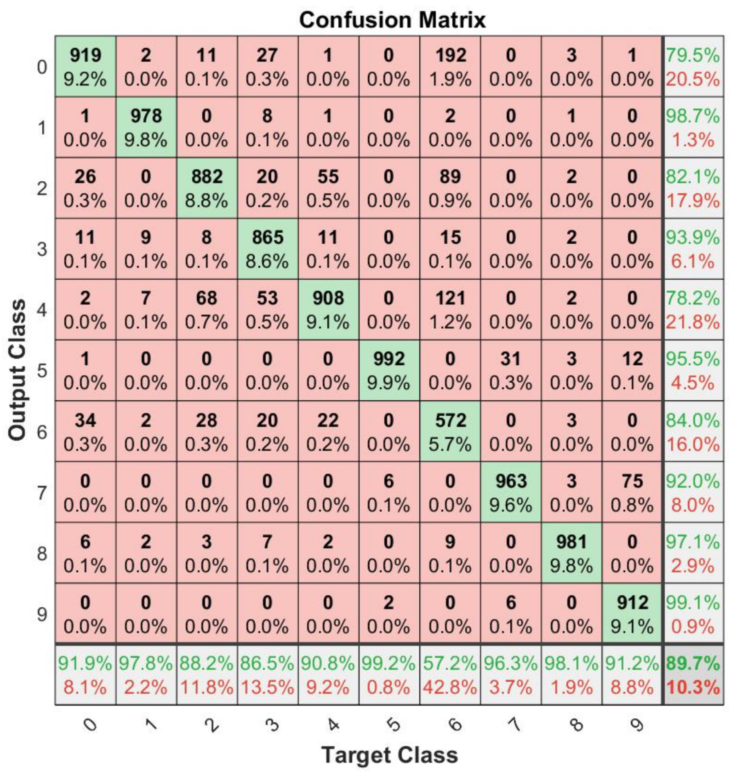

Confusion Matrix

Output Class \ Target Class	0	1	2	3	4	5	6	7	8	9	
0	807 8.1%	2 0.0%	22 0.2%	20 0.2%	0 0.0%	0 0.0%	95 0.9%	0 0.0%	4 0.0%	0 0.0%	84.9% 15.1%
1	2 0.0%	973 9.7%	1 0.0%	30 0.3%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	96.5% 3.5%
2	11 0.1%	0 0.0%	748 7.5%	9 0.1%	60 0.6%	0 0.0%	58 0.6%	0 0.0%	3 0.0%	0 0.0%	84.1% 15.9%
3	26 0.3%	15 0.1%	4 0.0%	822 8.2%	10 0.1%	0 0.0%	19 0.2%	0 0.0%	4 0.0%	0 0.0%	91.3% 8.7%
4	9 0.1%	5 0.1%	106 1.1%	70 0.7%	838 8.4%	0 0.0%	67 0.7%	0 0.0%	6 0.1%	0 0.0%	76.1% 23.9%
5	1 0.0%	0 0.0%	1 0.0%	1 0.0%	0 0.0%	941 9.4%	0 0.0%	25 0.3%	4 0.0%	12 0.1%	95.5% 4.5%
6	134 1.3%	4 0.0%	115 1.1%	43 0.4%	90 0.9%	0 0.0%	744 7.4%	0 0.0%	5 0.1%	1 0.0%	65.5% 34.5%
7	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	35 0.4%	0 0.0%	949 9.5%	4 0.0%	42 0.4%	92.0% 8.0%
8	10 0.1%	1 0.0%	2 0.0%	5 0.1%	2 0.0%	2 0.0%	15 0.1%	0 0.0%	970 9.7%	0 0.0%	96.3% 3.7%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	22 0.2%	0 0.0%	26 0.3%	0 0.0%	945 9.4%	95.2% 4.8%
	80.7% 19.3%	97.3% 2.7%	74.8% 25.2%	82.2% 17.8%	83.8% 16.2%	94.1% 5.9%	74.4% 25.6%	94.9% 5.1%	97.0% 3.0%	94.5% 5.5%	87.4% 12.6%

Figure 2: the confusion matrix of neural network with only fully-connected layers

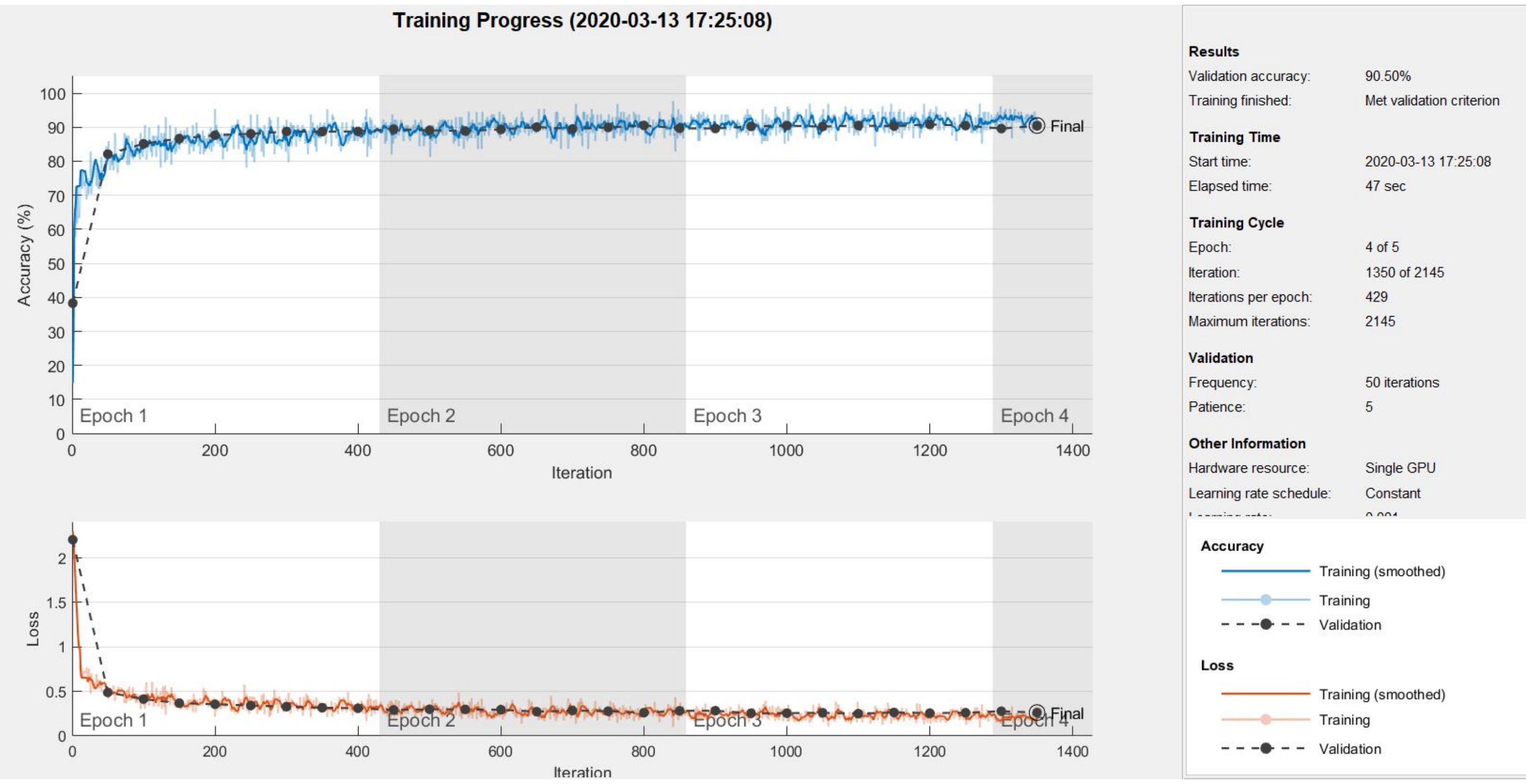
Part 2

In this part, we will use convolutional layers and fully-connected layers to construct our neural network. For the depth and width of our neural network, we choose two convolutional 2d layers with batch normalization layers and relu layers. Also, there is one fully connected layer with parameter 100. The validation accuracy is 90.50% as it shown in figure 4. And the confusion matrix is figure 3. From figure 2, we know that this model behave relatively bad in recognizing item 4 and item 0.


Confusion Matrix

Output Class \ Target Class	0	1	2	3	4	5	6	7	8	9	
0	919 9.2%	2 0.0%	11 0.1%	27 0.3%	1 0.0%	0 0.0%	192 1.9%	0 0.0%	3 0.0%	1 0.0%	79.5% 20.5%
1	1 0.0%	978 9.8%	0 0.0%	8 0.1%	1 0.0%	0 0.0%	2 0.0%	0 0.0%	1 0.0%	0 0.0%	98.7% 1.3%
2	26 0.3%	0 0.0%	882 8.8%	20 0.2%	55 0.5%	0 0.0%	89 0.9%	0 0.0%	2 0.0%	0 0.0%	82.1% 17.9%
3	11 0.1%	9 0.1%	8 0.1%	865 8.6%	11 0.1%	0 0.0%	15 0.1%	0 0.0%	2 0.0%	0 0.0%	93.9% 6.1%
4	2 0.0%	7 0.1%	68 0.7%	53 0.5%	908 9.1%	0 0.0%	121 1.2%	0 0.0%	2 0.0%	0 0.0%	78.2% 21.8%
5	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	992 9.9%	0 0.0%	31 0.3%	3 0.0%	12 0.1%	95.5% 4.5%
6	34 0.3%	2 0.0%	28 0.3%	20 0.2%	22 0.2%	0 0.0%	572 5.7%	0 0.0%	3 0.0%	0 0.0%	84.0% 16.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	6 0.1%	0 0.0%	963 9.6%	3 0.0%	75 0.8%	92.0% 8.0%
8	6 0.1%	2 0.0%	3 0.0%	7 0.1%	2 0.0%	0 0.0%	9 0.1%	0 0.0%	981 9.8%	0 0.0%	97.1% 2.9%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	6 0.1%	0 0.0%	912 9.1%	99.1% 0.9%
	91.9% 8.1%	97.8% 2.2%	88.2% 11.8%	86.5% 13.5%	90.8% 9.2%	99.2% 0.8%	57.2% 42.8%	96.3% 3.7%	98.1% 1.9%	91.2% 8.8%	89.7% 10.3%

Figure 3: the confusion matrix of neural network with convolutional and fully connected layers



Results
Validation accuracy: 90.50%
Training finished: Met validation criterion

Training Time
Start time: 2020-03-13 17:25:08
Elapsed time: 47 sec

Training Cycle
Epoch: 4 of 5
Iteration: 1350 of 2145
Iterations per epoch: 429
Maximum iterations: 2145

Validation
Frequency: 50 iterations
Patience: 5

Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant

Figure 4: the training process of neural networks with convolutional and fully connected layers

Summary and Conclusions

In this report , we explore the basic application of neural network, especially multilayer neural network. Although we do not have sufficient knowledge about how each hyperparameter work and affect the final result, we still successfully construct a relatively accurate model to classify items from MNIST dataset. From this process, we recognized the choice of appropriate number of layers and depth of neural network is essential for accuracy. Too many layers will not only overload GPU, but also extremely decrease the validation accuracy. Also, the convolutional layers is useful to increase accuracy of classification task. The comparison between different layers and different structures of neural network would help us to further understand the application of neural networks.

Appendix A

I2 = im2double(I) converts the image I to double precision. I can be a grayscale intensity image, a truecolor image, or a binary image. im2double rescales the output from integer data types to the range [0, 1].

B = reshape(A,sz) reshapes A using the size vector, sz, to define size(B). For example, reshape(A,[2,3]) reshapes A into a 2-by-3 matrix. sz must contain at least 2 elements, and prod(sz) must be the same as numel(A).

B = permute(A,dimorder) rearranges the dimensions of an array in the order specified by the vector dimorder. For example, permute(A,[2 1]) switches the row and column dimensions of a matrix A.

B = categorical(A) creates a categorical array from the array A. The categories of B are the sorted unique values from A.

layer = fullyConnectedLayer(outputSize) returns a fully connected layer and specifies the OutputSize property.

layer = reluLayer creates a ReLU layer.

layer = convolution2dLayer(filterSize,numFilters) creates a 2-D convolutional layer and sets the FilterSize and NumFilters properties.

layer = batchNormalizationLayer creates a batch normalization layer.

layer = maxPooling2dLayer(poolSize) creates a max pooling layer and sets the PoolSize property.

layer = softmaxLayer creates a softmax layer.

layer = classificationLayer creates a classification layer.

options = trainingOptions(solverName,Name,Value) returns training options with additional options specified by one or more name-value pair arguments.
[YPred,scores] = classify(net,tbl) predicts class labels for the data in tbl using the trained network, net.

plotconfusion(targets,outputs) plots a confusion matrix for the true labels targets and predicted labels outputs. Specify the labels as categorical vectors, or in one-of-N (one-hot) form.

description from https://www.mathworks.com/help/matlab/

Appendix B

```
clear all; close all; clc;
load ('fashion_mnist.mat');
% for k = 1 : 9
%     subplot(3,3,k)
%     imshow(reshape(X_train(k,:,:),[28,28]));
% end
X_train = im2double(X_train);
X_test = im2double(X_test);

X_train = reshape(X_train, [60000 28 28 1]);
X_train = permute(X_train, [2 3 4 1]);

X_test = reshape(X_test, [10000 28 28 1]);
X_test = permute(X_test, [2 3 4 1]);


X_valid = X_train(:,:,:,1:5000);
X_train = X_train(:,:,:,5001:end);

y_valid = categorical(y_train(1:5000))';
y_train = categorical(y_train(5001:end))';
y_test = categorical(y_test)';

% layers = [
%     imageInputLayer([28 28 1])
%
%     fullyConnectedLayer(392)
%     reluLayer
%     fullyConnectedLayer(196)
%     reluLayer
%     fullyConnectedLayer(10)
%     softmaxLayer
%     classificationLayer];


layers = [
    imageInputLayer([28 28 1])
    convolution2dLayer(5,16,'Padding',1)
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(5,32,'Padding',1)
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(100)
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];

options = trainingOptions('adam',...,
    'MaxEpochs',5,...
    'InitialLearnRate',1e-3,...
    'L2Regularization',1e-4,...
    'ValidationData',{X_valid,y_valid},...
    'Verbose',false,...
    'Plots','training-progress');

net = trainNetwork(X_train,y_train,layers,options);
```

```
y_pred = classify(net,X_train);
plotconfusion(y_train,y_pred);

figure(2);
y_pred = classify(net,X_valid);
plotconfusion(y_valid,y_pred);

y_pred = classify(net,X_test);
figure(3)
plotconfusion(y_test,y_pred);
saveas(gcf,'CNN_predicate.png');
```