# AMATH 482 Homework 5: Neural Networks for Classifying Fashion MNIST

Rose Ju

March 13, 2020

**Abstract**

The purpose of this project is to explore the data set, Fashion-MNIST, with a classifier. I will be using two neural networks to classify the images in the data set. The two types of neural networks are fully-connected neural network and convolutional neutral network. We will be experimenting several different neural network architectures with different hyperparameters and try to achieve higher accuracies. The factors that will be taken into consider, but not limited to, are the depth of the network, the width of the layers, the learning rates and many more.

## I. Introduction and Overview

In this project, I will be attempting to build an accurate classifier for the Fashion-MNIST data set. There are training images and test images for each category within the data. The goal is to correctly identify the category of a test image. I will be training neutral networks to classify the images within the data, Fully-connected neural network and convolutional neural network.

For the first part, I will use fully-connected neural network to classify the images. I will be adjusting its different architectures with different hyperparameters to achieve higher accuracies.

The adjustments include, but not limited to, the depth of the network, the width of the layer, the learning rate, and etc. I will try to achieve the best accuracy on the validation data with experimented adjustments. Next part, I will be applying the same procedure as the first part. But instead of using fully-connected neural network, I will be training the images with convolutional neural network. After calculating each parts accuracy, I will be comparing the results from each method.

# II. Theoretical Background

## i. Fully-Connected Neural Network

A fully connected neural network consists of a series of fully connected layers. It connects every nodes or "neuron" in one layer to every neuron in another layer. An advantage of fully connected neural network is structure agnostic, or no extra assumptions needs to be made about the data. In additional to the fully connected layer, the neural network also has a Relu layer. A Relu dense layer with Relu activation function. However, a downside of this method is having a large number of paraments such as slower training times and chances of overfitting.

## ii. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of deep neural networks. It is similar to the Fully-Connected Neural Network, but a regularized version. As mentioned above, one of the disadvantages of fully connected network is overfitting. CNN will apply the hierarchical pattern in data and assembles complex patterns using smaller patterns. Therefore, it will be less complex than the other methods. A CNN will consist a few components. An input layer consists an image's dissentions. A convolution layer that performs a convolution operation with parts of the input matrix. The sum of products of the corresponding elements is the output of the layer. A Relu dense layer with Relu activation function. Pooling layer to reduce the spatial size. Lastly, is the fully connected layer which is the same as the layer in fully connected neural network. With the repeated number of times,$n$, in performing each layer, it will form the $n$ deep neural network.

# III. Algorithm Implementation and Development

- Load the MNIST data set and convert the images to double precision.

- Reshape and reorder the arrays in training set and test set so that they have dimensions 60000×28×28 × 1 and 10000 × 28 × 28 × 1, respectively. remove 5000 images from the training data to use as validation data.
- Build the Fully-connected neutral network layer and compile the model
- Train the model by feeding the training data. Test the model with the test set and verify the predictions.
- Feed the model and evaluate its accuracy.
- Adjust the factors and paraments inside the layer or structure of the classifier to achieve higher accuracy until satisfaction
- Plot a graph to examine the relationship of the model loss and accuracy
- Repeat the process for convolutional neural network with the same steps. Except for step 3 build the additional layers, input layer, convolution layer, tanh layer and pooling layer.
- Compare the results for both methods. Evaluate their accuracies.

# IV. Computational Results

## i. Fully-Connected Neural Network

After performing the fully connect neural network classification, I find that the validation accuracy stabilizes around 89.09%. As shown in figure 1, I perform with 5 epochs and the accuracy eventually stabilizes along with the entropy loss kept below 0.5. I adjusted the learning

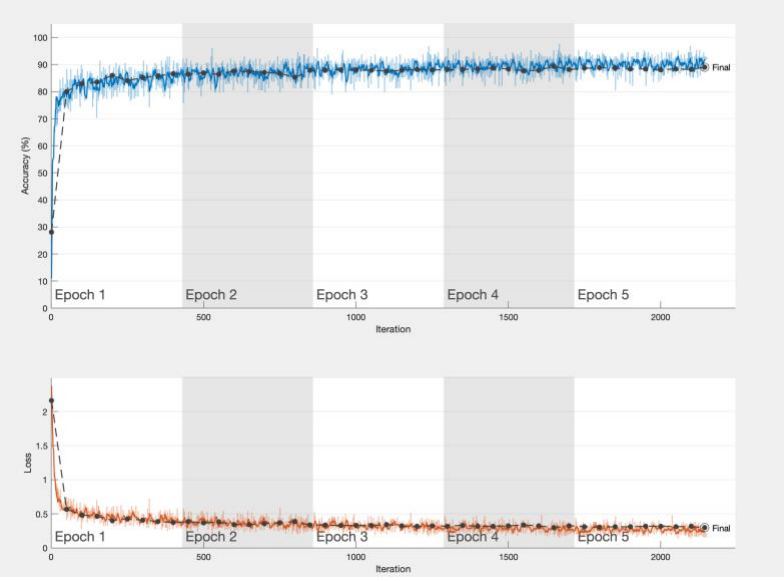


*Figure 1:Training Progress for Accuracy and Entropy Loss with respect to each Iteration*

rate and the regularization parameter to 7e-4 and 1e-7 for maximizing accuracies. Also, while

exploring for accuracy maximization, I found that changing the depth of the network and the width of the layer do not affect the accuracy by a significant amount. In addition, I calculated the confusion matrix for the training set and test classifier as shown below in figure 2 and figure 3. I found the training set accuracy in the confusion matrix to be approximately 3% higher than the test classifier. The training set seems to have a 91.3% accuracy where the final test classifier has an accuracy of 88.4%.

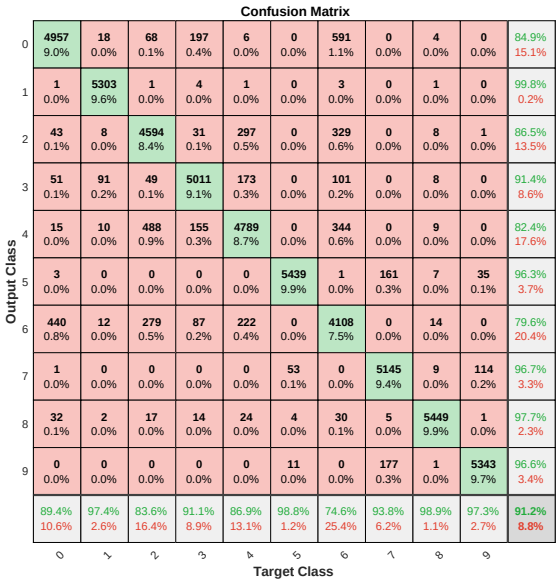


*Figure 2: Confusion Matrix for Training set with Fully Connected Neutral network*

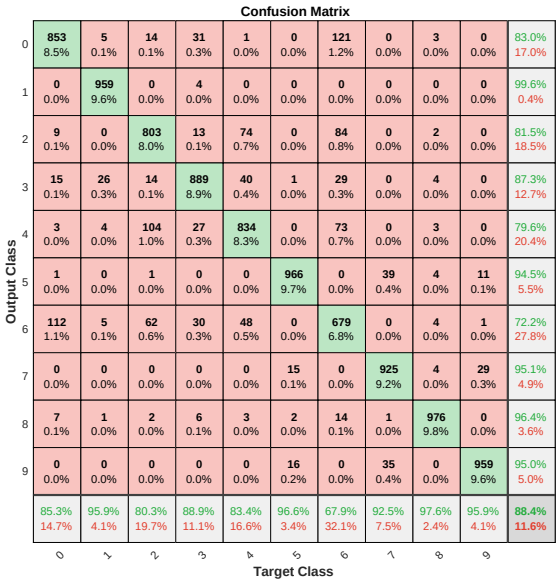


*Figure 3: Test Classifier with the testing set with CNN*

## ii. Convolutional Neural Network

After performing the convolutional neural network classification, I find that the validation accuracy stabilizes around 87.94%. As shown in figure 4, I perform the training process with 5 epochs and the accuracy eventually stabilizes along with the entropy loss kept below 0.5. Similar to the Fully-Connected adjustments, the learning rate and the regularization parameter was set to 7e-4 and 1e-7 for maximizing accuracies. I set up the convolutional layer, tanh layer, pooling layer and the fully connected layer. I calculated the confusion matrix for the training set and test classifier as shown below in figure 5 and figure 6. I found the training set accuracy in the confusion matrix to be approximately 1% higher than the test classifier. The training set seems to have a 88.7% accuracy where the final test classifier has an accuracy of 87.2%.

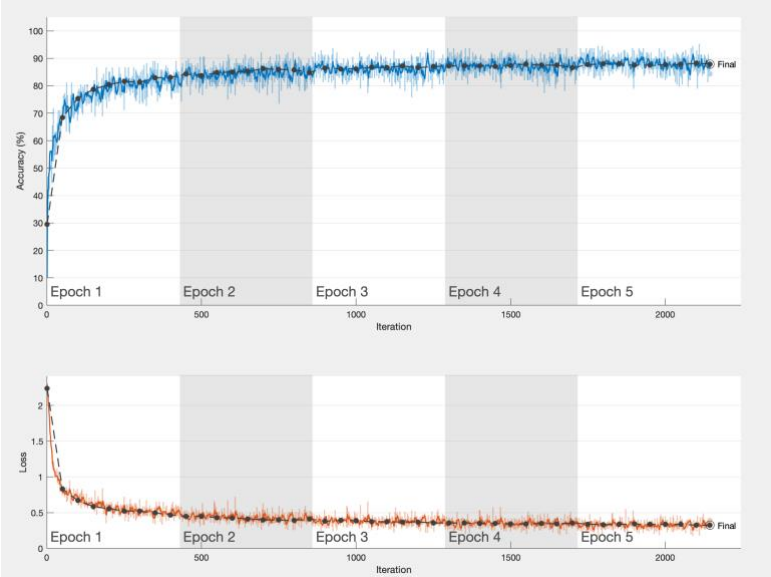


*Figure 4: Training Progress for Accuracy and Entropy Loss with respect to each Iteration*

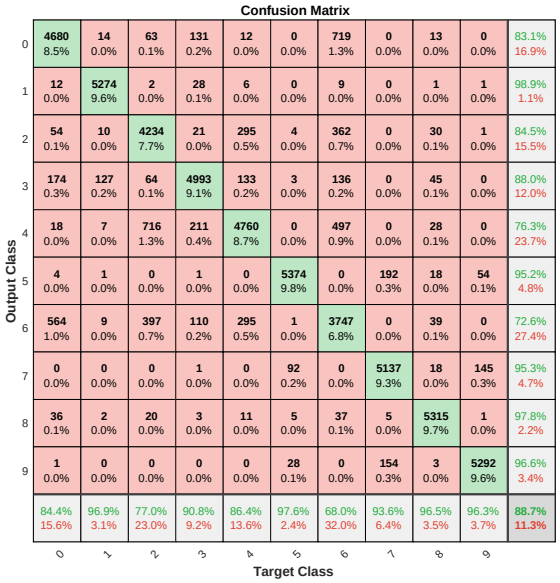


*Figure 5: Confusion Matrix for Training set with CNN*

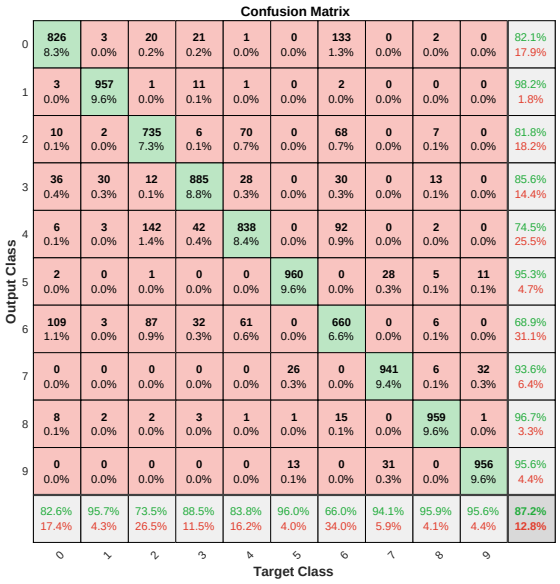


*Figure 6: Test Classifier with the testing set with CNN*

## V. Summary and Conclusions

I build the classifier with both fully connected neutral network and the convolutional neutral network. We found from the training progress that the validation accuracy for fully connect NN and CNN to be 89.09% and 87.94% respectively. With the built classifier, we find the final test classifier for the fully connected neutral network to be 88.4%. On the other side, the convolutional neutral network's accuracy is slightly lower, 87.2%. The final test accuracy seems to be very similar from both classifiers. If more number trials could run, the mean accuracy for

all the trials could be even closer to each other. I can tell from the confusion matrix and the test classifier that there shows no sign of overfitting. For future improvements for better accuracy, this experiments for building a classifier could increase the number of filters in the convolutional layer. Perhaps look into the number of epoch as well.

# VI. Appendix A – MATLAB functions

**`averagePooling2dLayer(X)`** – constructs the pooling layer of the neural network. It is used to reduce the spatial size

**`classificationLayer(X)`** – constructs the classification layer of the neural network.

**`convolution2dLayer (X)`** – constructs the convolutional layer of the neural network. It is used to extract features from the images in the dataset.

**`fullyConnectedLayer(X)`** – constructs the fully connected layer of the neural network. each parameter is linked to one another to determine the true relation and effect of each parameter on the labels.

**`imageinputlayer(X)`** – constructs the input layer of the neural network.

**`reluLayer(X)`** – constructs the relu layer of the neural network. A Relu dense layer with Relu activation function

**`softmaxLayer(X)`** – constructs the softmax layer of the neural network.

**`trainingsOptions(X)`** – Create a set of options for training a network. Including InitialLearnRate, L2Regularization, ValidationData, plots and more

## VII. Appendix B – MATLAB codes

```
%% MNIST Classifier with deep neural network
clear; close all; clc

load('mnist.mat')

figure(1)
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(300)
    reluLayer
    fullyConnectedLayer(100)
    reluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];

options = trainingOptions('adam', ...
    'MaxEpochs',5,...
    'InitialLearnRate',7e-4, ...
    'L2Regularization',1e-7, ...
    'ValidationData',{X_valid,y_valid}, ...
    'Verbose',false, ...
    'Plots','training-progress');

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

%% Confusion for training
figure(2)
y_pred = classify(net,X_train);
plotconfusion(y_train,y_pred)

%% Test classifier
```

```
figure(3)
y_pred = classify(net,X_test);
plotconfusion(y_test,y_pred)
%% MNIST Classifier with convolutional neural network
clear; close all; clc

load('mnist.mat')

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],"Name","imageinput")
    convolution2dLayer([5 5],6,"Name","conv_1","Padding","same")
    tanhLayer("Name","tanh_1")
    averagePooling2dLayer([2
2],"Name","avgpool2d_1","Padding","same","Stride",[2 2])
    convolution2dLayer([5 5],16,"Name","conv_2")
    tanhLayer("Name","tanh_3")
    averagePooling2dLayer([2
2],"Name","avgpool2d_2","Padding","same","Stride",[2 2])
    convolution2dLayer([5 5],120,"Name","conv_3")
    tanhLayer("Name","tanh_2")
    fullyConnectedLayer(84,"Name","fc_1")
    tanhLayer("Name","tanh_4")
    fullyConnectedLayer(10,"Name","fc_2")
    softmaxLayer("Name","softmax")
    classificationLayer("Name","classoutput")];

options = trainingOptions('adam', ...
    'MaxEpochs',5,...
    'InitialLearnRate',7e-4, ...
    'L2Regularization',1e-7, ...
    'ValidationData',{X_valid,y_valid}, ...
    'Verbose',false, ...
    'Plots','training-progress');

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

%% Confusion for training
figure(1)
y_pred = classify(net,X_train);
```

```
plotconfusion(y_train,y_pred)

%% Test classifier
figure(2)
y_pred = classify(net,X_test);
plotconfusion(y_test,y_pred)
```