Neural Networks for Classifying Fashion MNIST

Eileen Chang

March 13, 2020

Abstract

In this paper, we are going build two networks. One is fully-connected neural network, the other is convolutional neural network. Before training, we have to prepare the data, such as convert it to double, reshape, etc. Then, we can start to train the model. Our goal is trying to make the accuracy as high as possible. In order to achieve that, we can modify some factors, for example, the number of the training layers, the number of the neurons, etc.

1 Introduction and Overview

We are going to explore algorithms of image classification - the "Fully-Connected Neural Network" and the "Convolutional Neural Network (CNN)." With the right dataset and training, these models should be able to make predictions and classify images on levels similar to human ability. We will be using the "Fashion-MNIST" dataset to train and test our models. Our aim is to achieve the highest level of accuracy from our models. We can manipulate the accuracy through different factors, such as the initial learning rate, L2 Regularization, etc.

2 Theoretical Background

2.1 Fully-Connected Neural Network

Fully-connected Neural Network is an artificial neural network that be able to interpret the data by going through a layer or multiple layers. We take the values in the previous layer and multiply them by some weights then plug into the activation function. These new values are the numbers in each neuron of the next layer. The number of neurons in each layer and the activation function are the important factors that will affect the accuracy of the model. The aim of this algorithm is to try to behave as humans' brains. It helps us cluster and classifies the data. The principle behind this algorithm is it is trying to simulate the working process of humans' brains. The patterns they recognize are numerical, contained in vectors.

2.2 Convolutional Neural Network

Convolutional Neural Network is the process before doing a fully-connected neural network. It does the convolution and max pooling on the input data. Take the image for example. The reason for doing convolution is it wants to make each pixel in the picture related to the pixels around it. That is, convolution will weight each pixel and sum up them to become a new pixel for a new matrix. And the main idea behind a pooling layer is to "accumulate" features from maps generated by convolving a filter over an image.

3 Algorithm Implementation and Development

Down below is the algorithm for both fully-connected neural network and CNN.

- Load the data into Matlab.
- Convert train data and test data to double precision.
- Reshape the training data to 60000*28*28*1 and the test data to 10000*28*28*1.
- Permute the dimensions of data.
- Split the training data into valid data(5000) and training data(remain).
- Assign the labels to categorizes.
- Data is ready.

This is the algorithm for fully-connected neural network.

- Build the training network.
    1. imageInputLayer
    2. fullyConnectedLayer with 522 hidden neurons
    3. reluLayer
    4. fullyConnectedLayer with 261 hidden neurons
    5. reluLayer
    6. fullyConnectedLayer with 10 neurons
    7. SoftmaxLayer
    8. ClassificationLayer
- Set the hyper-parameters for the training network.
    1. Adam
    2. MiniBatchSize(64)
    3. MaxEpochs(15)
    4. InitialLearnRate(1e-3)

5. L2Rugularization(5e-8)
6. ValidationData(XValid, YValid)
7. ValidationFrequency(30)
8. LearnRateSchedule(piecewise)
9. GradientDecayFactor(0.7)
10. Verbose(false)

This is the algorithm for Convolutional Neural Network.

- Build the training network.
  1. imageInputLayer
  2. convolution2dLayer(32 filters with 5*5)
  3. reluLayer
  4. averagePooling2dLayer(2*2 with stride 2)
  5. convolution2dLayer(64 filters with 5*5)
  6. reluLayer
  7. averagePooling2dLayer(2*2 with stride 2)
  8. convolution2dLayer(128 filters with 5*5)
  9. reluLayer
  10. averagePooling2dLayer(2*2 with stride 2)
  11. fullyConnectedLayer with 128 hidden neurons
  12. reluLayer
  13. fullyConnectedLayer with 10 neurons
  14. SoftmaxLayer
  15. ClassificationLayer
- Set the hyper-parameters for the training network.
  1. SGDM
  2. MaxEpochs(15)
  3. InitialLearnRate(9e-2)
  4. L2Rugularization(5e-4)
  5. ValidationData(XValid, YValid)
  6. ValidationFrequency(200)
  7. Verbose(false)
- Then assign training data, labels, layers, and hyper-parameters into function trainNetwork.
- Finally, test the model with train data and test data.

4 Computational Results

4.1 Task 1 - Fully-Connected Neural Network

In order to find the best hyper-parameter, I change the number of the fully-ConnectedLayer. Then, I get 0.85 accuracy for the training process. However, our goal is about 0.9. So, I start to change the initial learning rate. I raise it from 1e-3 to 1e-5. It did perform better but not as good as 0.9. Then, I change the L2Regularization from 1e-3 to 1e-7. It booths up a little bit, but still not good enough. After that, I change the number of neurons in each layer of fully-ConnectedLayer. The aim of this process is to smooth down from 28*28 to 10. I also modify the max epochs, validation frequency, gradient decay factor, the optimizer, and the activation function to make the model better.

After hitting the walls so many times, figure 1 is the best performance.

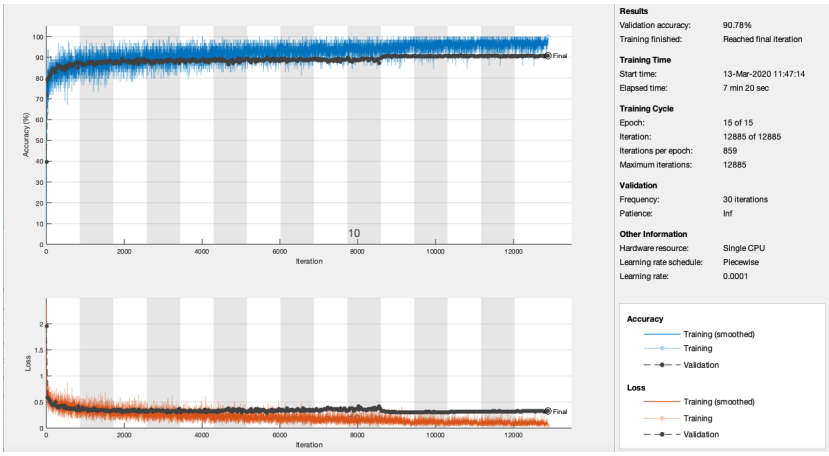

Figure 1: The Training Process of Fully-Connected Neural Network.

Figure 1 is the training process of the fully-connected neural network. The total epoch is 15, and the validation frequency is 30. The initial learning rate is 0.001. From this graph, we can see that it booths up a little bit when it hit the 11 epochs. The accuracy in the training process is 0.9078.

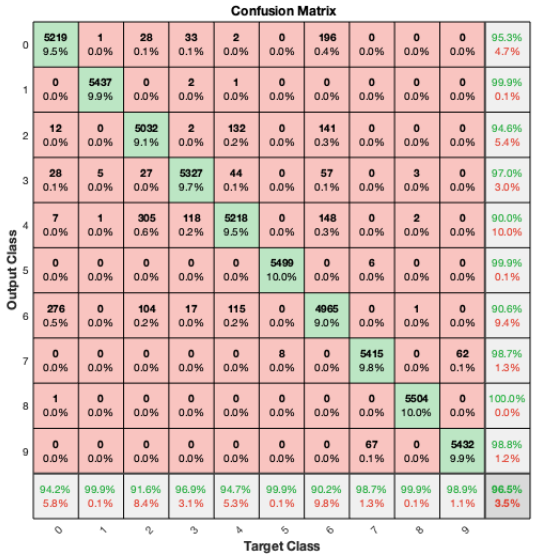

Figure 2: Confusion Matrix with Training Data for Task 1.

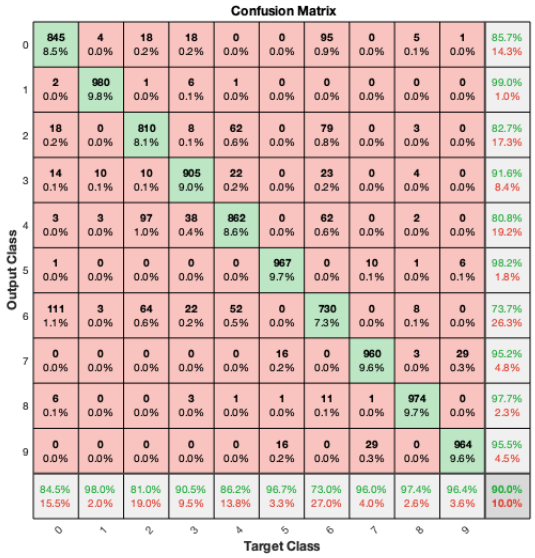

Figure 3: Confusion Matrix with Testing Data for Task 1.

From figure 2, we can tell that it fits the training data really well, about 0.96 success rate. From figure 3, we have the success rate of 0.9 for the testing data.

4.2 Task 2 - Convolutional Neural Network

First I try only one layer, it performs not bad but not as good as 0.9. It only has about 0.8. So, I add one more convolution layer in it. Then, I try the activation function with relu since from the previous task, I find out that relu seems to perform better than other functions for these data set. Also, I try to modify the filters in each convolution layer and the number of neurons for the hidden layer. Moreover, I have experienced the max epochs, initial learning rate, L2Regularization, Validation frequency, the optimizer, and the size and stride of the max pooling.

After hitting the walls so many times, figure 4 is the best performance.

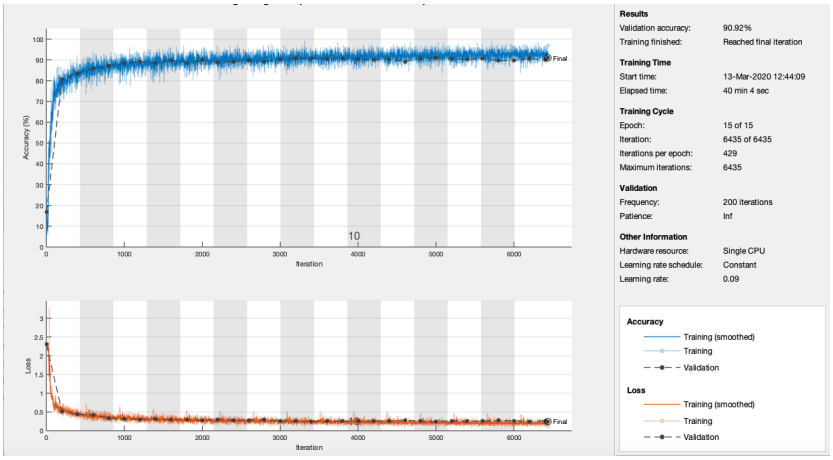

Figure 4: The Training Process of Convolutional Neural Network.

Figure 4 is the training process of the convolutional neural network. The total epoch is 15, and the validation frequency is 200. The initial learning rate is 0.09. It takes really long time(40 min). If I set the smaller validation frequency, it will take longer than this. The accuracy in the training process is 0.9092.

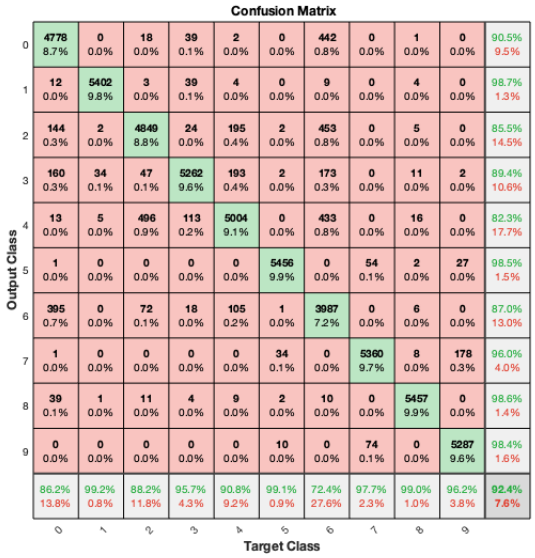

Figure 5: Confusion Matrix with Training Data for Task 2.

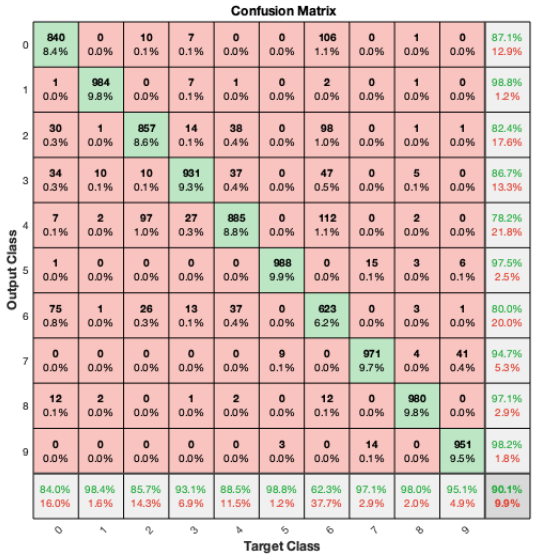

Figure 6: Confusion Matrix with Testing Data for Task 2.

From figure 5, we can tell that it fits the training data really well, about 0.924 success rate. From figure 6, we have the success rate of 0.901 for the testing data.

5 Summary and Conclusions

It is very challenging to find the hyper-parameter that makes a high prediction because there are a lot of factors that will change the results. From the experience, the performance between the two algorithms is about the same. The highest success rate for fully-connected neural network and convolutional neural network are 0.9 and 0.901 respectively. However, the fully-connected neural network has higher accuracy when putting the training data into the model to test. It has about 0.965. And the convolutional neural network only have 0.924. The convolutional neural network takes much longer to run compared with fully-connected neural network. It takes about 40 min for CNN. And fully-connected neural network only takes 7 min.

6 Appendix A.

1. Machine Learning Tool
   - layers - the place to build the network.
   - trainingOptions - modify the factors to change the success rate.
2. im2double
   - Convert image to double precision
   - I2 = im2double(I) converts the image I to double precision.
3. permute
   - Permute array dimensions.
   - B = permute(A,dimorder) rearranges the dimensions of an array in the order specified by the vector dimorder.
4. categorical
   - Array that contains values assigned to categories.
   - B = categorical(A) creates a categorical array from the array A. The categories of B are the sorted unique values from A.

7 Appendix B.

```
%% MNIST Classifier with fully-connected neural network
clc; close all; clear all;
load fashion_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])
    fullyConnectedLayer(522)
    reluLayer
    fullyConnectedLayer(261)
    reluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
options = trainingOptions('adam', ...
    'MiniBatchSize', 64, ...
    'MaxEpochs',15,...
    'InitialLearnRate',1e-3, ...
    'L2Regularization',5e-8, ...
    'ValidationData',{X_valid,y_valid}, ...
    'ValidationFrequency', 30, ...
    'LearnRateSchedule','piecewise', ...
    'GradientDecayFactor', 0.7, ...
    '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 fashion_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],32,"Name","conv_1","Padding","same")
    reluLayer("Name","relu_1")
    averagePooling2dLayer([2 2],"Name","avgpool2d_1","Padding","same","Stride",2
    convolution2dLayer([5 5],64,"Name","conv_2")
    reluLayer("Name","relu_3")
    averagePooling2dLayer([2 2],"Name","avgpool2d_2","Padding","same","Stride",2
    convolution2dLayer([5 5],128,"Name","conv_3")
    reluLayer("Name","relu_2")
    averagePooling2dLayer([2 2],"Name","avgpool2d_3","Padding","same","Stride",2
    fullyConnectedLayer(128,"Name","fc_1")
    reluLayer("Name","relu_4")
    fullyConnectedLayer(10,"Name","fc_2")
    softmaxLayer("Name","softmax")
    classificationLayer("Name","classoutput")];
options = trainingOptions('sgdm', ...
    'MaxEpochs',15,...
    'InitialLearnRate',9e-2, ...
    'L2Regularization',5e-4, ...
    'ValidationData',{X_valid,y_valid}, ...
    'Verbose',false, ...
    'ValidationFrequency', 200, ...
    '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)
```