Jeremy Aguirre

AMATH 482

March 13, 2020

Assignment #5: Neural Networks for Classifying Fashion MNIST

Abstract

In this assignment, a MNIST fashion data set is used for the training, validation, and testing of neural networks. A fully connected neural network and a convolutional neural network are developed, and different hyper parameters are adjusted to determine their effects on the networks' accuracy in classifying test data. The fully connected neural network was 86.84% accurate in classifying the training data during training and 85.7% accurate in classifying the test data. The convolutional neural network was 88.86% accurate in classifying the validation data during training and 87.9% accurate in classifying the test data.

I. Introduction and Overview

In this assignment, a MNIST fashion data set is used for the training, validation, and testing of neural networks. A fully connected neural network and a convolutional neural network are developed, and different hyper parameters are adjusted to determine their effects on the networks' accuracy in classifying test data.

II. Theoretical Background

VGGnet is a deep convolutional network developed for large scale visual recognition by the Visual Geometry Group at the University of Oxford. Its layout is as follows:

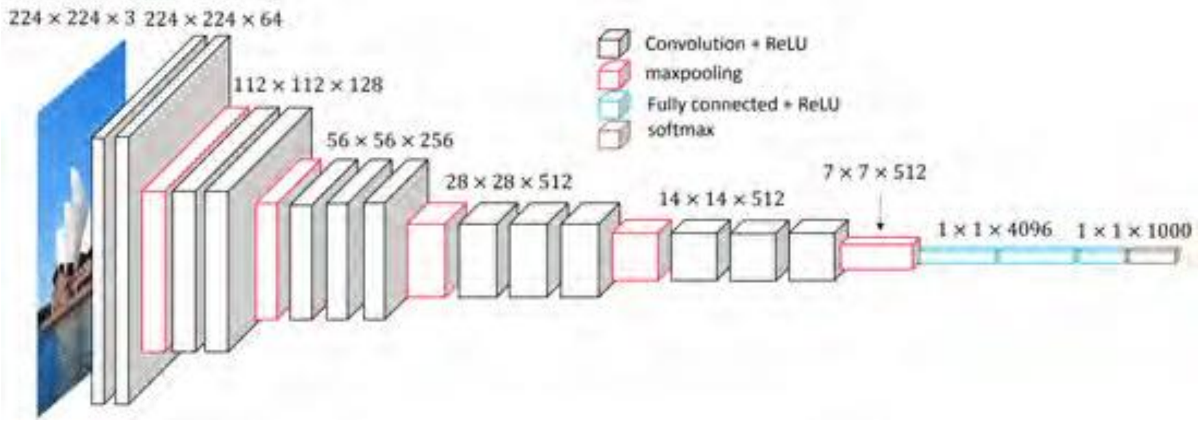


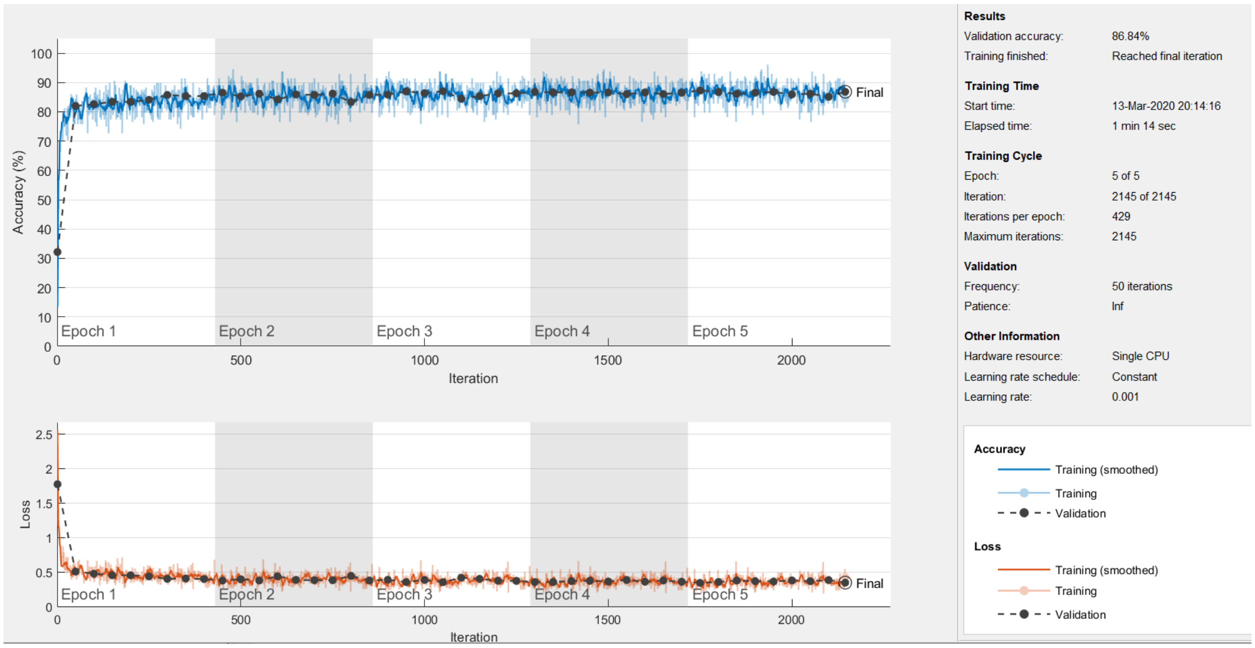
Figure 1. VGGnet architecture

III. Algorithm Implementation and Development

A fashion MNIST data set was provided in a .mat format. The data was converted into double precision format using the im2double command. Using the reshape and permute commands, the data then had to be converted into a format compatible with the MATLAB Deep Learning toolbox. The first 5000 figures in the training data were then extracted to be used as validation data. The labels included in the data set also had to be formatted using the command categorical for them to be formatted properly. Then, using the Deep Network Designer app in MATLAB a fully connected neural network was constructed. It consisted of two hidden layers; the first one was a fully connected layer with 500 outputs and a hyperbolic tangent activation function. The second fully connected layer had 300 outputs with a hyperbolic tangent activation function. The final layer had 10 outputs with the softmax activation function which correspond to the 10 classification types for the fashion MNIST. Training of the network with validation was performed using the built in MATLAB commands trainNetwork and trainingOptions. The training was performed using adaptive moment estimation across 5 epochs. The learning rate and the L2 regularization value were both set to 1e-3. Following the training of the fully connected neural network, confusion plots were made for the classification of the training and test data.

For the convolutional neural network, the same process was performed for the new network with different hyper parameters. The layout of this convolutional network was derivative of the VGGnet and is included in Appendix B.

IV. Computational Results



Results
Validation accuracy: 86.84%
Training finished: Reached final iteration

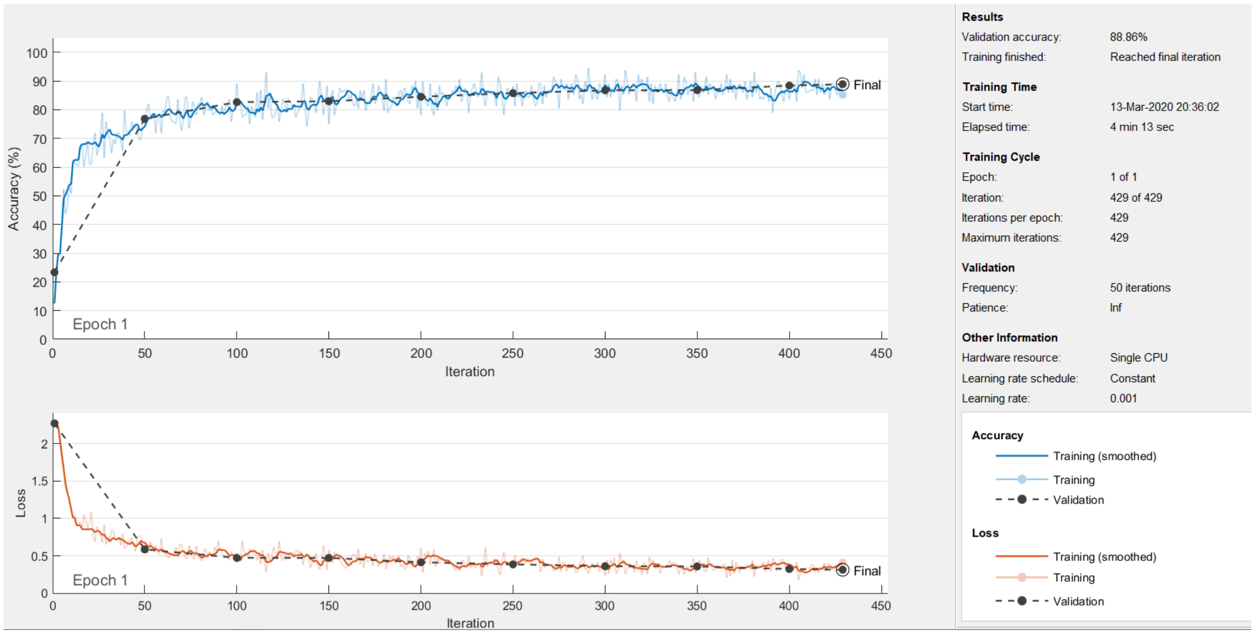
Training Time
Start time: 13-Mar-2020 20:14:16
Elapsed time: 1 min 14 sec

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

Validation
Frequency: 50 iterations
Patience: Inf

Other Information
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.001

Figure 2. Training accuracy and loss function for the fully connected neural network



Results

Validation accuracy:	88.86%
Training finished:	Reached final iteration

Training Time

Start time:	13-Mar-2020 20:36:02
Elapsed time:	4 min 13 sec

Training Cycle

Epoch:	1 of 1
Iteration:	429 of 429
Iterations per epoch:	429
Maximum iterations:	429

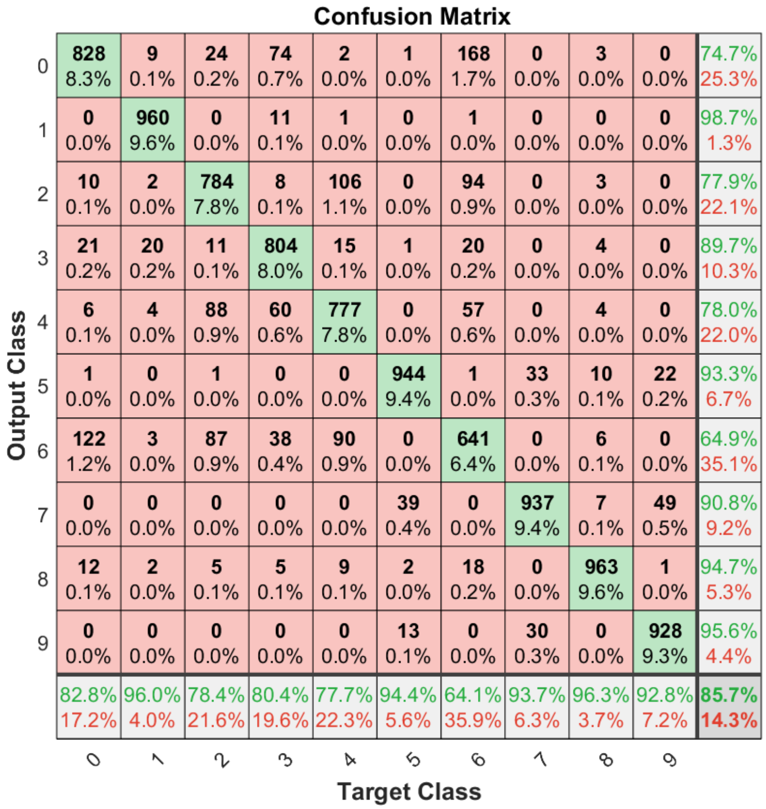
Validation

Frequency:	50 iterations
Patience:	Inf

Other Information

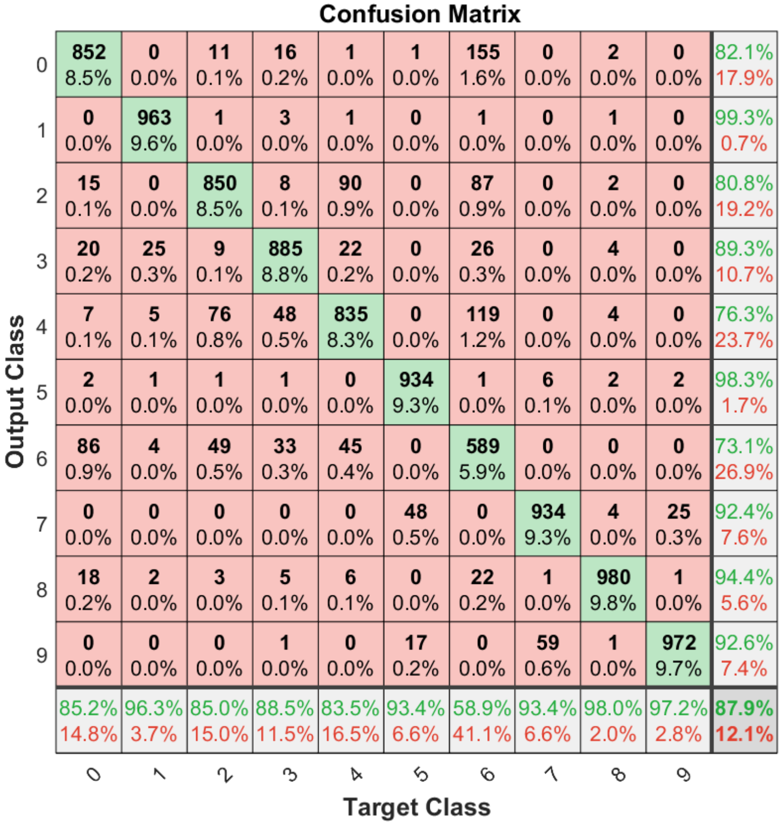
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.001

Figure 3. Validation accuracy and loss function for the convolutional neural network


Confusion Matrix
Output Class
Target Class

	0	1	2	3	4	5	6	7	8	9	
0	**828** 8.3%	**9** 0.1%	**24** 0.2%	**74** 0.7%	**2** 0.0%	**1** 0.0%	**168** 1.7%	**0** 0.0%	**3** 0.0%	**0** 0.0%	74.7% 25.3%
1	**0** 0.0%	**960** 9.6%	**0** 0.0%	**11** 0.1%	**1** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	98.7% 1.3%
2	**10** 0.1%	**2** 0.0%	**784** 7.8%	**8** 0.1%	**106** 1.1%	**0** 0.0%	**94** 0.9%	**0** 0.0%	**3** 0.0%	**0** 0.0%	77.9% 22.1%
3	**21** 0.2%	**20** 0.2%	**11** 0.1%	**804** 8.0%	**15** 0.1%	**1** 0.0%	**20** 0.2%	**0** 0.0%	**4** 0.0%	**0** 0.0%	89.7% 10.3%
4	**6** 0.1%	**4** 0.0%	**88** 0.9%	**60** 0.6%	**777** 7.8%	**0** 0.0%	**57** 0.6%	**0** 0.0%	**4** 0.0%	**0** 0.0%	78.0% 22.0%
5	**1** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**944** 9.4%	**1** 0.0%	**33** 0.3%	**10** 0.1%	**22** 0.2%	93.3% 6.7%
6	**122** 1.2%	**3** 0.0%	**87** 0.9%	**38** 0.4%	**90** 0.9%	**0** 0.0%	**641** 6.4%	**0** 0.0%	**6** 0.1%	**0** 0.0%	64.9% 35.1%
7	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**39** 0.4%	**0** 0.0%	**937** 9.4%	**7** 0.1%	**49** 0.5%	90.8% 9.2%
8	**12** 0.1%	**2** 0.0%	**5** 0.1%	**5** 0.1%	**9** 0.1%	**2** 0.0%	**18** 0.2%	**0** 0.0%	**963** 9.6%	**1** 0.0%	94.7% 5.3%
9	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**13** 0.1%	**0** 0.0%	**30** 0.3%	**0** 0.0%	**928** 9.3%	95.6% 4.4%
	82.8% 17.2%	96.0% 4.0%	78.4% 21.6%	80.4% 19.6%	77.7% 22.3%	94.4% 5.6%	64.1% 35.9%	93.7% 6.3%	96.3% 3.7%	92.8% 7.2%	**85.7%** **14.3%**

Figure 4. Confusion matrix for the classification of test data by the fully connected neural network



Output Class \ Target Class	0	1	2	3	4	5	6	7	8	9	
0	852 8.5%	0 0.0%	11 0.1%	16 0.2%	1 0.0%	1 0.0%	155 1.6%	0 0.0%	2 0.0%	0 0.0%	82.1% 17.9%
1	0 0.0%	963 9.6%	1 0.0%	3 0.0%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	99.3% 0.7%
2	15 0.1%	0 0.0%	850 8.5%	8 0.1%	90 0.9%	0 0.0%	87 0.9%	0 0.0%	2 0.0%	0 0.0%	80.8% 19.2%
3	20 0.2%	25 0.3%	9 0.1%	885 8.8%	22 0.2%	0 0.0%	26 0.3%	0 0.0%	4 0.0%	0 0.0%	89.3% 10.7%
4	7 0.1%	5 0.1%	76 0.8%	48 0.5%	835 8.3%	0 0.0%	119 1.2%	0 0.0%	4 0.0%	0 0.0%	76.3% 23.7%
5	2 0.0%	1 0.0%	1 0.0%	1 0.0%	0 0.0%	934 9.3%	1 0.0%	6 0.1%	2 0.0%	2 0.0%	98.3% 1.7%
6	86 0.9%	4 0.0%	49 0.5%	33 0.3%	45 0.4%	0 0.0%	589 5.9%	0 0.0%	0 0.0%	0 0.0%	73.1% 26.9%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	48 0.5%	0 0.0%	934 9.3%	4 0.0%	25 0.3%	92.4% 7.6%
8	18 0.2%	2 0.0%	3 0.0%	5 0.1%	6 0.1%	0 0.0%	22 0.2%	1 0.0%	980 9.8%	1 0.0%	94.4% 5.6%
9	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	17 0.2%	0 0.0%	59 0.6%	1 0.0%	972 9.7%	92.6% 7.4%
	85.2% 14.8%	96.3% 3.7%	85.0% 15.0%	88.5% 11.5%	83.5% 16.5%	93.4% 6.6%	58.9% 41.1%	93.4% 6.6%	98.0% 2.0%	97.2% 2.8%	**87.9%** **12.1%**

Figure 5. Confusion matrix for the classification of test data by the convolutional neural network

The fully connected neural network was 86.84% accurate in classifying the training data during training and 85.7% accurate in classifying the test data. When classifying the test data, the fully connected neural network had particular difficulty differentiating between t-shirts and shirts, as well as confusing pullovers, coats, and shirts.

The convolutional neural network was 88.86% accurate in classifying the validation data during training and 87.9% accurate in classifying the test data. In classifying the test data, the convolutional neural network also had difficulty identifying coats, pullovers, shirts, and t-shirts.

V. Summary and Conclusions

A fully connected neural network and a convolutional neural network were developed to classify fashion MNIST. There were a number of hyper parameters which were adjusted in order to increase the accuracy of these networks. Increasing the depth of each network as well as the width of their layers seemed to slightly improve accuracy after a period of time in training, however, these changes had to be accompanied by a decrease in the learning rate in order for the loss function to continue to display desired behavior. The downside to this change was that training took longer. Adjusting the regularization parameter seemed to have a larger effect on the accuracy of each neural network, but it had to be carefully tuned after any change to other hyper parameters. Having the regularization parameters set too high or too low in a given network configuration is detrimental to its accuracy. In testing different activation functions, the hyperbolic tangent function consistently yielded higher accuracy for my networks. The relu activation function only performed noticeably less. The adaptive moment estimation optimization technique outperformed the stochastic gradient descent with momentum and root mean square propagation techniques consistently. For the hyper parameters specific to the convolutional neural network, changes had to be more nuanced due to its complexity, however, I generally found that increasing the number of filters, decreasing filter size and strides, smaller pool sizes, and padding layers increased accuracy.

Appendix A

Deep Network Designer App – This app is included with the MATLAB Deep Learning Toolbox and it provides a graphical interface for creating, training, and simulating shallow and deep learning neural networks.

Appendix B

```
%% MNIST Fashion Classifier with Fully-Connected Neural Network
clear all; close all; clc
load('fashion_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(500)
        tanhLayer
```

```
    fullyConnectedLayer(300)
    tanhLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
options = trainingOptions('adam', ...
  'MaxEpochs',5, ...
  'InitialLearnRate',1e-3, ...
  'L2Regularization', 1e-3, ...
  'ValidationData',{X_valid,y_valid}, ...
  'Verbose',false, ...
  'Plots','training-progress');
net = trainNetwork(X_train,y_train,layers,options);

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

%% Test Confusion
figure(3)
y_pred = classify(net,X_test);
plotconfusion(y_test,y_pred)
print('-f1','hw51','-dpng')
print('-f2','hw52','-dpng')
print('-f3','hw53','-dpng')

%% MNIST Fashion 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],64,"Name","conv_1","Padding","same")
```

```
   reluLayer("Name","relu_1")
   maxPooling2dLayer([2 2],"Name","maxpool_1","Padding","same","Stride",[2 2])
   convolution2dLayer([5 5],128,"Name","conv_2")
   reluLayer("Name","relu_2")
   maxPooling2dLayer([2 2],"Name","maxpool_2","Padding","same","Stride",[2 2])
   convolution2dLayer([5 5],256,"Name","conv_3")
   reluLayer("Name","relu_3")
   maxPooling2dLayer([2 2],"Name","maxpool_3","Padding","same","Stride",[2 2])
   fullyConnectedLayer(4096,"Name","fc_1")
   reluLayer("Name","relu_6")
   fullyConnectedLayer(10,"Name","fc_2")
   softmaxLayer("Name","softmax")
   classificationLayer("Name","classoutput")];
options = trainingOptions('adam', ...
   'MaxEpochs',1,...
   'InitialLearnRate',1e-3, ...
   'L2Regularization',1e-3, ...
   'ValidationData',{X_valid,y_valid}, ...
   'Verbose',false, ...
   'Plots','training-progress');
net = trainNetwork(X_train,y_train,layers,options);

%% Training Confusion
figure(4)
y_pred = classify(net,X_train);
plotconfusion(y_train,y_pred)

%% Test Confusion
figure(5)
y_pred = classify(net,X_test);
plotconfusion(y_test,y_pred)
print('-f4','hw54','-dpng')
print('-f5','hw55','-dpng')
```