Table of Contents


```
clc; clear all; close all;
```

Delivery report Pt 2

This report contains the awnsers to questions posed in the Deliverable

Deliverable 3ci

```
% The number of neurons that worked better for step 3  was 400 for a
 sample
% size of 9000. This number of neurons worked better because
 increasing the
% number of neurons showed a decrese in accuracy and favorable
 readings
% from the ROC (Receiver Operating Characteristic). This is most
 commanly
% because of overfitting to the training set of the data.
```

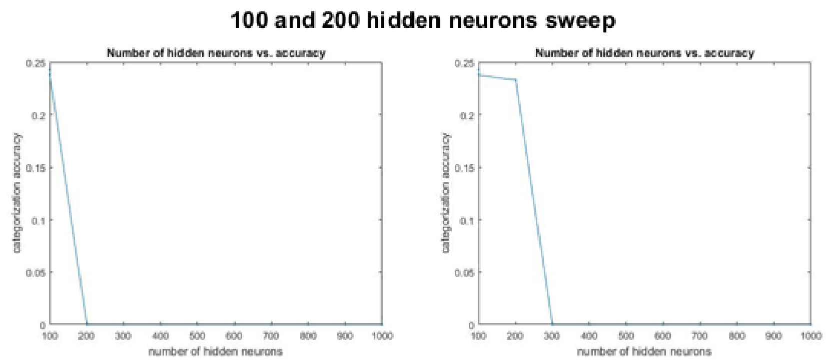
Deliverable 3cii

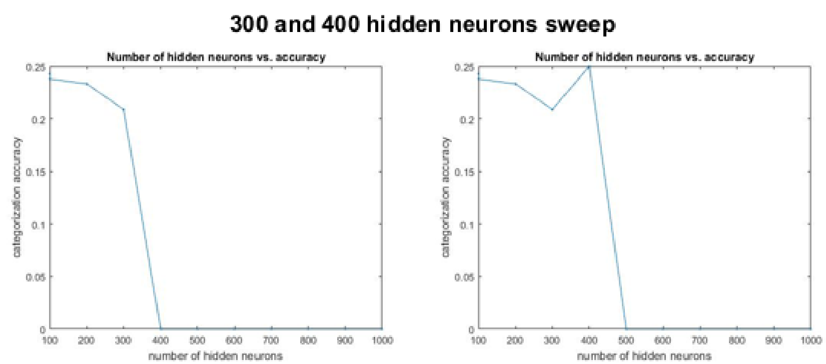
```
%The extraction method that worked better was the one with less inputs
 and
%more information. The extraction of the frequency magnatude
 components
%showed an increased accuraccy at different levels of neurons in the
 sweep.
%This was because the features that were fed to the input had more
%information in them for the neural network.
```

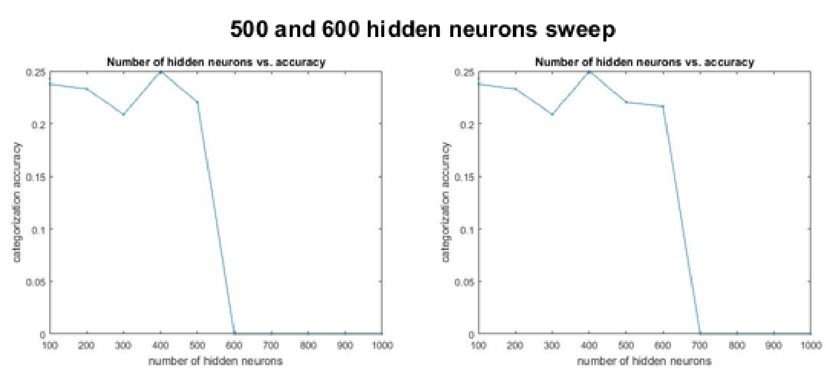
Part 3 Training a machine to understand emtion (The Sweep)

```
%Let it be known that the whole training dataset was not used in
 training
%this neural network. It was modified to only take 15,000 training
 samples
%and use 9,000 of it to sweep for the number of hidden neurons.
```

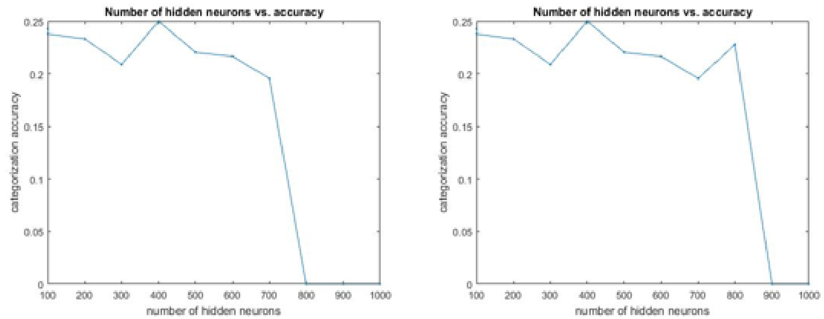
```
image1 = imresize(imread('./Final/100.jpg'), 0.5);
image2 = imresize(imread('./Final/200.jpg'), 0.5);
image3 = imresize(imread('./Final/300.jpg'), 0.5);
image4 = imresize(imread('./Final/400.jpg'), 0.5);
image5 = imresize(imread('./Final/500.jpg'), 0.5);
image6 = imresize(imread('./Final/600.jpg'), 0.5);
image7 = imresize(imread('./Final/700.jpg'), 0.5);
image8 = imresize(imread('./Final/800.jpg'), 0.5);
image9 = imresize(imread('./Final/900.jpg'), 0.5);
image10 = imresize(imread('./Final/1000.jpg'), 0.5);
final = imread('./Final/1000.jpg');
plot = [image1 image2];
plot2 = [image3 image4];
plot3 = [image5 image6];
plot4 = [image7 image8];
plot5 = [image9 image10];
figure; imshow(plot); title('100 and 200 hidden neurons sweep');
figure; imshow(plot2); title('300 and 400 hidden neurons sweep');
figure; imshow(plot3); title('500 and 600 hidden neurons sweep');
figure; imshow(plot4); title('700 and 800 hidden neurons sweep');
figure; imshow(plot5); title('900 and 1000 hidden neurons sweep');
figure; imshow(final); title('Final System Accuracy');

%Below shows each sweep iteration from 100 to 1000, incrementing by
 100
%each time. The fluctuation that we are seeing is due to
%the fact that the neural network is being trained with a different
 number
%of neurons each iteration. As can be seen, the accuracy per neurons
%decreases until it reaches 400 where it spikes to a 25% accuracy and
 then
%continues to decrease until the final iteration. This means that
 training
%the neural network with 400 neurons would give us the most accurate
%outputs. The final system accuracy is shown below also. As you can
 see,
%400 neurons is the most accurate in our sweep.

image_p = imread('./Final/percentage.jpg');
figure; imshow(image_p); title('Accuracy after percentage change');

%We also manipulated the percentage of data going to training and
 testing
%in the sweep iteration for loop in the hopes that we would correct
%overfitting more. However, the results yielded less accuracy as seen
%below.
```


100 and 200 hidden neurons sweep
Number of hidden neurons vs. accuracy
categorization accuracy
number of hidden neurons
Number of hidden neurons vs. accuracy
categorization accuracy
number of hidden neurons

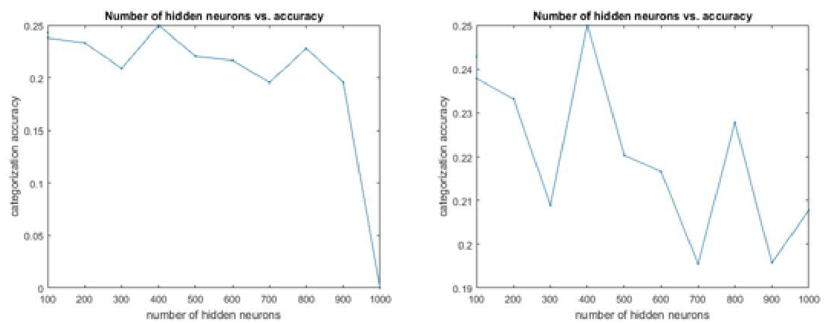

300 and 400 hidden neurons sweep
Number of hidden neurons vs. accuracy
categorization accuracy
number of hidden neurons
Number of hidden neurons vs. accuracy
categorization accuracy
number of hidden neurons

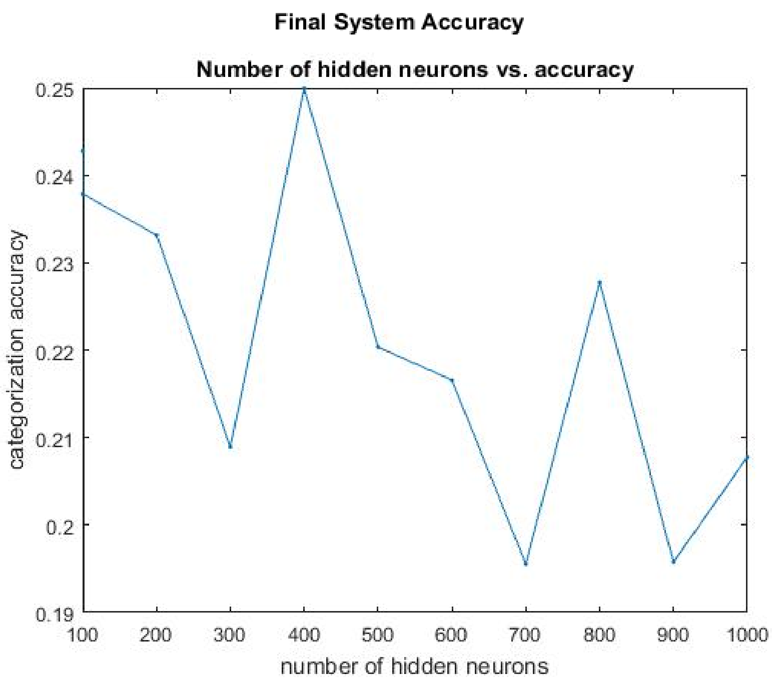

500 and 600 hidden neurons sweep
Number of hidden neurons vs. accuracy
categorization accuracy
number of hidden neurons
Number of hidden neurons vs. accuracy
categorization accuracy
number of hidden neurons

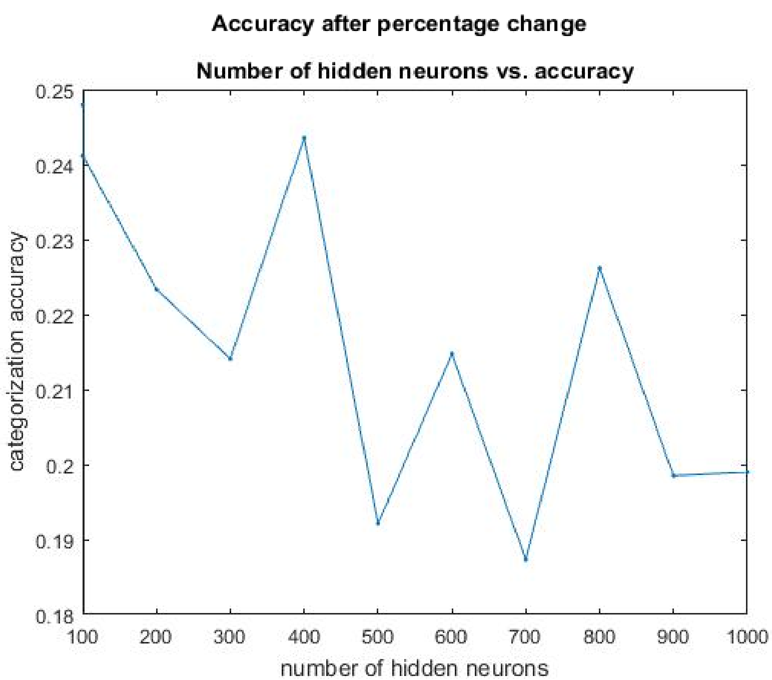
700 and 800 hidden neurons sweep



900 and 1000 hidden neurons sweep




Final System Accuracy
Number of hidden neurons vs. accuracy
categorization accuracy
number of hidden neurons
0.25
0.24
0.23
0.22
0.21
0.2
0.19
100
200
300
400
500
600
700
800
900
1000

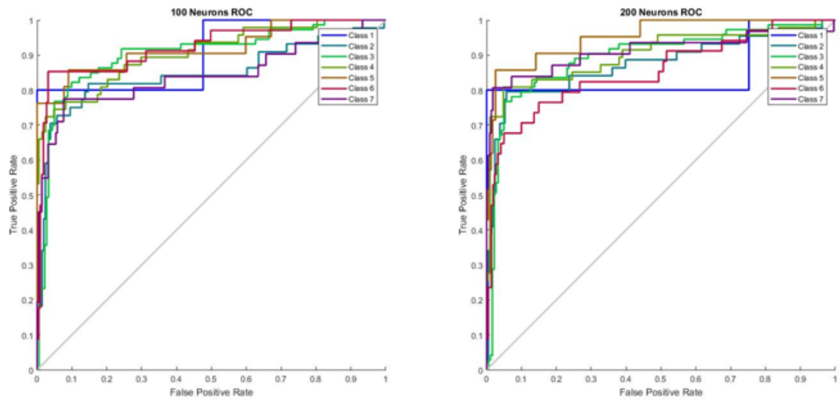

Accuracy after percentage change
Number of hidden neurons vs. accuracy
categorization accuracy
number of hidden neurons
0.25
0.24
0.23
0.22
0.21
0.2
0.19
0.18
100
200
300
400
500
600
700
800
900
1000

Part 3 ROC curves of each iteration

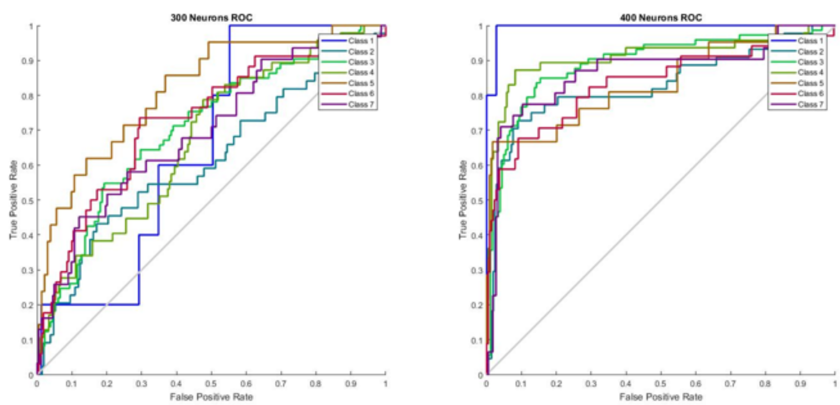
```
image1_n = imresize(imread('./Final/100 neurons.jpg'), 0.5);
image2_n = imresize(imread('./Final/200 neurons.jpg'), 0.5);
image3_n = imresize(imread('./Final/300 neurons.jpg'), 0.5);
image4_n = imresize(imread('./Final/400 neurons.jpg'), 0.5);
image5_n = imresize(imread('./Final/500 neurons.jpg'), 0.5);
image6_n = imresize(imread('./Final/600 neurons.jpg'), 0.5);
image7_n = imresize(imread('./Final/700 neurons.jpg'), 0.5);
image8_n = imresize(imread('./Final/800 neurons.jpg'), 0.5);
image9_n = imresize(imread('./Final/900 neurons.jpg'), 0.5);
image10_n = imresize(imread('./Final/1000 neurons.jpg'), 0.5);
final_n = imread('./Final/ROC_sweep.jpg');

plot_n = [image1_n image2_n];
plot2_n = [image3_n image4_n];
plot3_n = [image5_n image6_n];
plot4_n = [image7_n image8_n];
plot5_n = [image9_n image10_n];
figure; imshow(plot_n); title('100 and 200 hidden neurons ROC curve');
figure; imshow(plot2_n); title('300 and 400 hidden neurons ROC
 curve');
figure; imshow(plot3_n); title('500 and 600 hidden neurons ROC
 curve');
figure; imshow(plot4_n); title('700 and 800 hidden neurons ROC
 curve');
figure; imshow(plot5_n); title('900 and 1000 hidden neurons ROC
 curve');
figure; imshow(final_n); title('Final ROC curve');

%Each of the ROC curves shown below represent the performance of the
 neural
%network when being trained with the specified number of hidden
 neurons.
%Each emotion is represented by a class, as can be seen on each graph:
%(7=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral)
%What we are looking for is the ROC curve with the most classes
%located primarily in the upper left hand quadrant of the plot. This
 will
%indiacte that the neural networks performance, with respect to each
%class, is good.It can be seen that the plot utilizing 400 hidden
 neurons
%shows the best ROC curve.
%The final ROC curve is the overall ROC curve for the neural network.
```

Warning: Image is too big to fit on screen; displaying at 67%

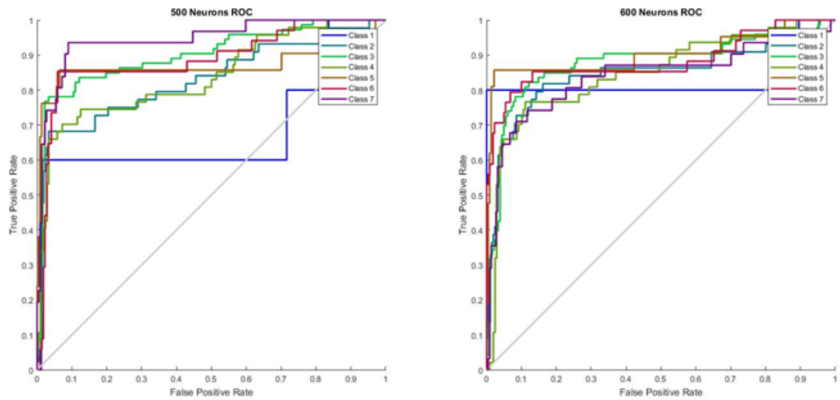
100 and 200 hidden neurons ROC curve



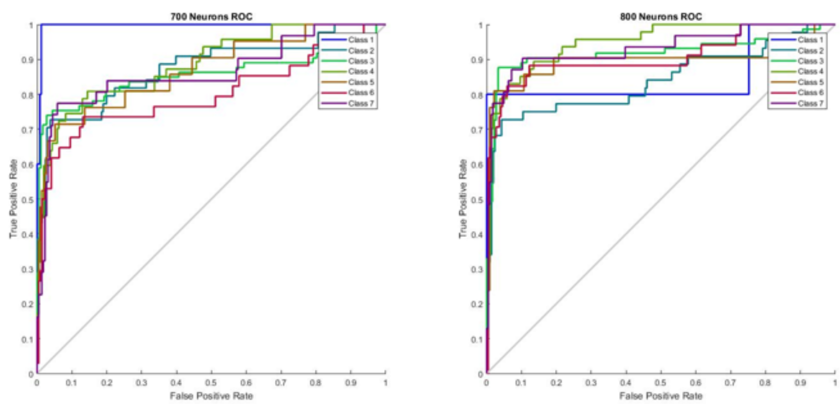
300 and 400 hidden neurons ROC curve



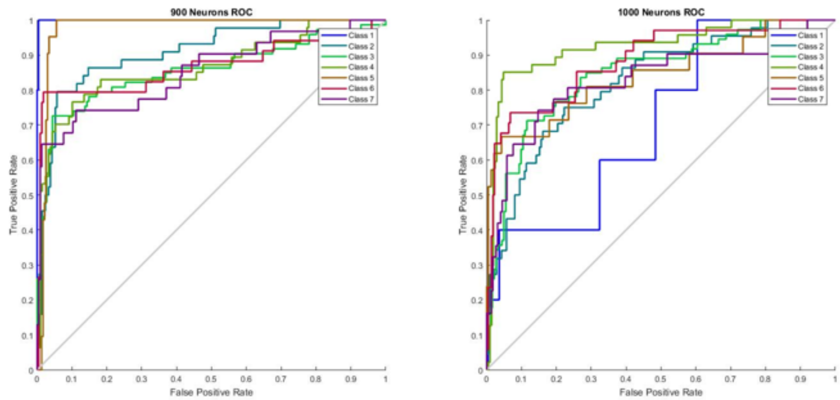
500 and 600 hidden neurons ROC curve



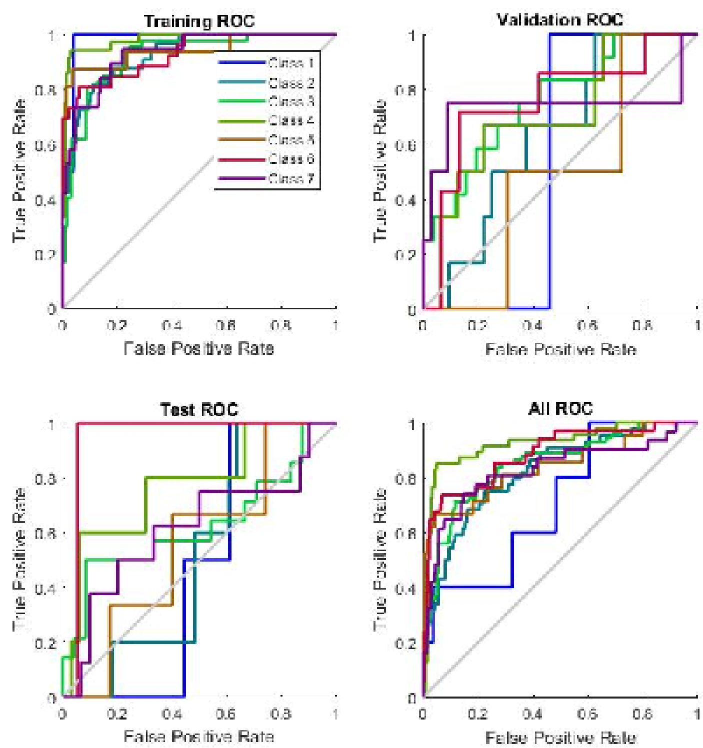
700 and 800 hidden neurons ROC curve



900 and 1000 hidden neurons ROC curve



Final ROC curve



Part 4 Methods to reduce complexity

```
%Three different methods that could be used to reduce the complexity
 of the
%system are:
%1. wavelet: using wavelet transform would take out a lot of the noise
 in
%the images, allowing the neural network to grab better pixel values
 from
%the data sets.
%2. frequency domain: by taking the frequency domain, we could take
 lower
%values of an image and create a more precise data set for training
 and
%testing
%3. downsize image: downsizing an image would decrease the size of the
%dataset which could decrease the chance of overfitting the training
 set.
%This could reduce the complexity of the training set and increase the
%accuracy.

%All three of the above methods would manipulate the input images and
%create a more precise training and testing set for the neural network
 to
%be trained with. This would create a better performing and more
 accurate
%neural network.
```

Part 5 Sweep ROC

```
sweep = [10,10:10:250];

for i = 1:21
    formatSpec = "./Q5figSaves/N%dRoc";
    savefigpath = sprintf(formatSpec,sweep(i));
    openfig(savefigpath);

end
% close all
```

```
Error using openFigure
The value of 'Filename' is invalid. It must satisfy the function:
 ischar.

Error in openfig>localGetFileAndOptions (line 98)
ip.parse(args{:});

Error in openfig (line 37)
[filename, reuse, visibleAction] = localGetFileAndOptions(varargin);

Error in report2 (line 130)
    openfig(savefigpath);
```

Published with MATLAB® R2017a

Table of Contents


```
clear;
clc;

close all;     %closes all figures
```

data formatting

fer2013.csv - training data test.csv - test data for submission

```
disp('loading data ...............');

% The training set is  28709 samples
% testing set is  7178  samples
fer = fopen('fer2013.csv', 'r');                    % read fer2013.csv
tr = 28709;
testSize = 7178;
fullSize = tr + testSize;

%get headers and data from dataset
headers = textscan(fer, '%s %s %s', 1, 'delimiter', ', ');
data = textscan(fer, '%d %s %s', fullSize, 'delimiter', ',');
```

```
loading data ...............
```

all data for training

```
%defining all arrays for the grabbing the data
pixels = [];
emotions =[];
trainingPixels = [];
testPixels = [];
testEmotions =[];
testingPixels = [];

%parse out training values for emotions and pixels
```

```
for i=1:15000
    pixels = [pixels; data{1,2}(i)];                  %parsing all pixel
 values
    emotions = [emotions; data{1,1}(i)];              %parsing all
 emotion values
    stringPix = char(pixels{i,1});                    %convert into
 string
    parsePix = str2double(strsplit(stringPix));       %seperate each
 pixel value
    trainingPixels = [trainingPixels; uint8(emotions(i,1)),
 uint8(parsePix)]; %put all into new training array
end
```

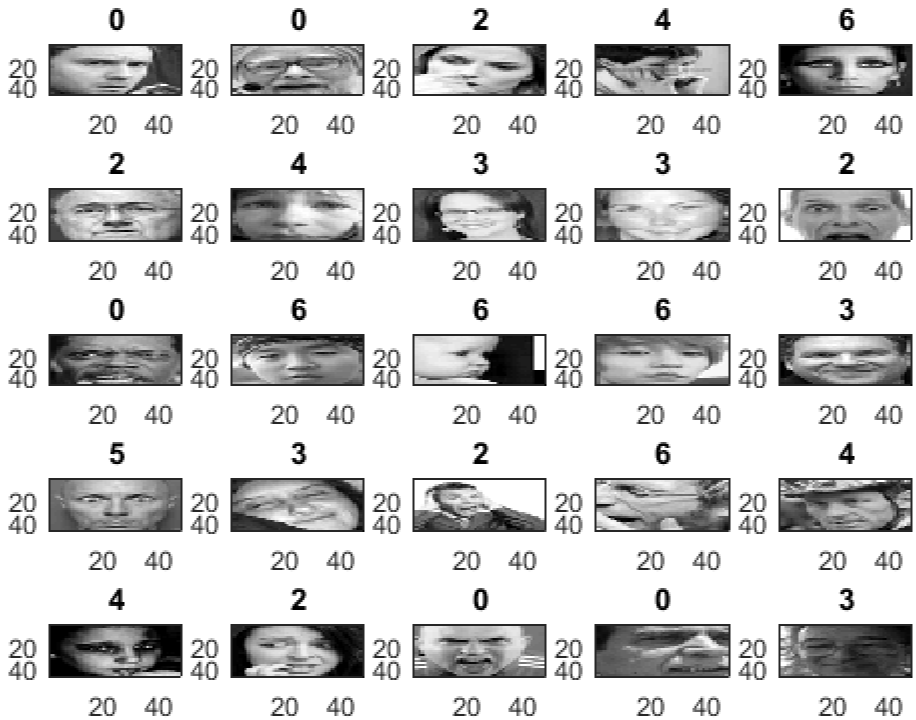
all data for testing

```
%parse out testing data
for j=tr+1:fullSize
    testPixels = [testPixels; data{1,2}(j)];              %parsing all
 pixel values
    testEmotions = [testEmotions; data{1,1}(j)];          %parsing all
 emotion values
end
%parse out each testing pixel we need
for k=1:testSize
    stringTestPix = char(testPixels{k,1});                %convert into
 string
    parseTestPix = str2double(strsplit(stringTestPix)); %seperate each
 pixel value
    testingPixels = [testingPixels; uint8(testEmotions(k,1)),
 uint8(parseTestPix)]; %put all new values into a new testing array
end



disp('Loaded ....');
```

Reshape the data to Visualize example for the digits sample

```
figure    ;                                                   % plot
 images
colormap(gray)                                                % set to
 grayscale
for i = 1:25                                                  % preview
 first 25 samples
    subplot(5,5,i)                                            % plot
 them in 6 x 6 grid
    digit = reshape(trainingPixels(i, 2:end), [48,48])';    % row = 48
 x 48 image
    imagesc(digit)                                            % show the
 image
```

```
    title(num2str(trainingPixels(i, 1)))                    % show the
 label
end
```



The dataset stores samples in rows rather than in columns, so you need to

transpose it. Then you will partition the data so that you hold out 1/3 of the data for model evaluation, and you will only use 2/3 for training our artificial neural network model.

```
% n = size(trainingPixels, 1);                        % number of samples
 in the dataset
n = 9000;
targets  = double(trainingPixels(:,1));                     % 1st column
 is |label|
targets(targets == 0) = 7;                           % use '7' to present
 '0'
targetsd = dummyvar(targets);                        % convert label into a
 dummy variable

% No need for the first column in the (trainingPixels) set any longer
inputs = double(trainingPixels(:,2:end));                   % the rest of
 columns are predictors; have to double so all inputs are the same

inputs = inputs';                  % transpose input
```

```
targets = targets';                          % transpose target
targetsd = targetsd';                        % transpose dummy variable
```

partitioning the dataset based on random selection of indices

```
rng(1);                                                       % for
 reproducibility
patitionObject = cvpartition(n,'Holdout', uint8(n/3));    % hold out
 1/3 of the dataset

Xtrain = inputs(:, training(patitionObject));     % 2/3 of the input
 for training
Ytrain = targetsd(:, training(patitionObject));  % 2/3 of the target
 for training

Xtest = inputs(:, test(patitionObject));          % 1/3 of the input
 for testing
Ytest = targets(test(patitionObject));            % 1/3 of the target
 for testing
Ytestd = targetsd(:, test(patitionObject));       % 1/3 of the dummy
 variable for testing


disp('Ready for NNstart...');
```

Ready for NNstart...

Time to Run the Neural Network GUI Application

```
% type NNstart on the command prompt
```

Computing the Categorization Accuracy

```
Ypred = myNNfun(Xtest);                      % predicts probability for each
 label
Ypred(:, 1:5)                                % display the first 5 columns
[~, Ypred] = max(Ypred);                     % find the indices of max
 probabilities
sum(Ytest == Ypred) / length(Ytest); % compare the predicted vs.
 actual
```

```
ans =

    0.0042    0.0008    0.0000    0.0001    0.0000
    0.0013    0.2113    0.9811    0.5026    0.0001
    0.0008    0.3507    0.0189    0.0056    0.0001
    0.9607    0.0002    0.0000    0.1572    0.0000
```

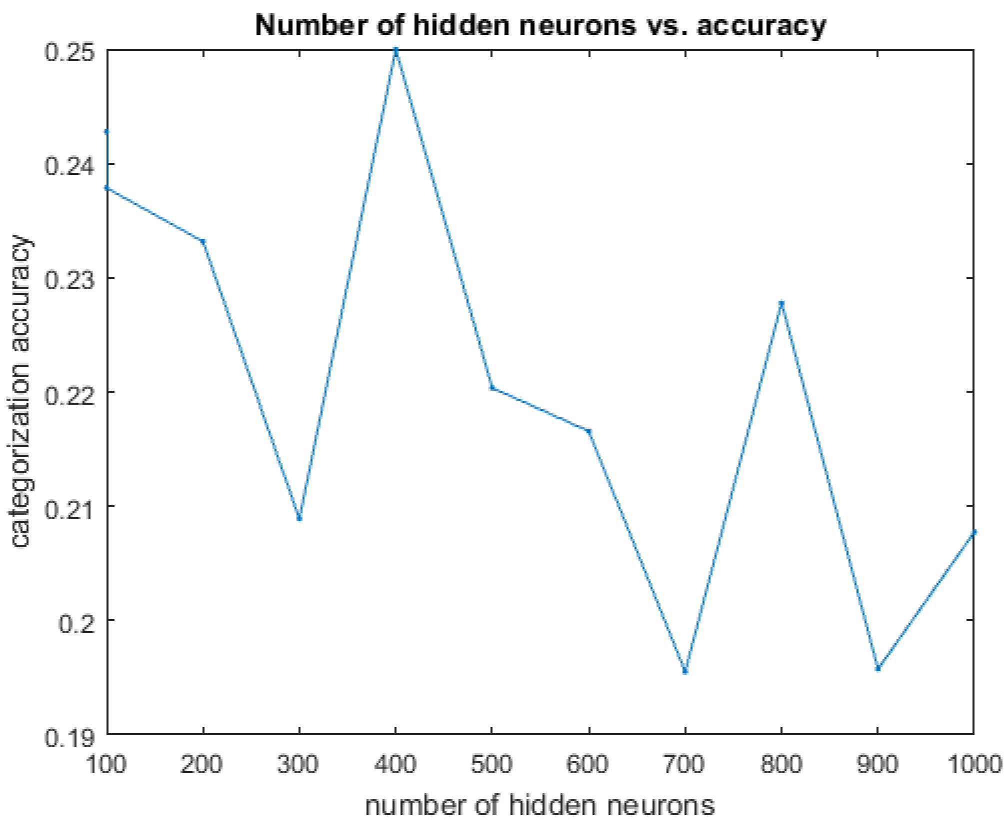
```
    0.0023    0.0410    0.0001    0.0007    0.0208
    0.0268    0.0001    0.0000    0.3337    0.9790
    0.0039    0.3958    0.0000    0.0001    0.0000
```

Sweep Code Block

Sweeping to choose different sizes for the hidden layer

```
sweep = [100,100:100:1000];                  % parameter values to test
scores = zeros(length(sweep), 1);           % pre-allocation
% we will use models to save the several neural network result from
 this
% sweep and run loop
models = cell(length(sweep), 1);         % pre-allocation
x = Xtrain;                              % inputs
t = Ytrain;                              % targets
trainFcn = 'trainscg';                   % scaled conjugate gradient

%    figure
for i = 1:length(sweep)
    hiddenLayerSize = sweep(i);          % number of hidden layer
 neurons
    net = patternnet(hiddenLayerSize);  % pattern recognition network
    net.divideParam.trainRatio = 70/100;% 70% of data for training
    net.divideParam.valRatio = 15/100;  % 15% of data for validation
    net.divideParam.testRatio = 15/100; % 15% of data for testing
    net = train(net, x, t);              % train the network
%     net = train(net, x, t,'useParallel','yes');
 % ,'useGPU','yes','showResources','yes'train the network
%
%
%     simpleclusterOutputs = sim(net,x);
%
%     % Ploting the ROC
%     plotroc(t,simpleclusterOutputs,sprintf('%d Neurons' ,sweep(i)));
%
%     formatSpec = "./Q5figSaves/N%dRoc";
%     savefigpath = sprintf(formatSpec,sweep(i));
%   pause();


    models{i} = net;                     % store the trained network
    p = net(Xtest);                      % predictions
    [~, p] = max(p);                     % predicted labels
    scores(i) = sum(Ytest == p) /length(Ytest);  % categorization
 accuracy
```

```
%     plot(sweep, scores, '.-')
% xlabel('number of hidden neurons')
% ylabel('categorization accuracy')
% title('Number of hidden neurons vs. accuracy')
%     pause();



end
% Let's now plot how the categorization accuracy changes versus number
 of
% neurons in the hidden layer.

figure
plot(sweep, scores, '.-')
xlabel('number of hidden neurons')
ylabel('categorization accuracy')
title('Number of hidden neurons vs. accuracy')
```



Published with MATLAB® R2017a

Table of Contents

Feature Set Up

This section will go throught the steps to extract some features that will be used to train our new NN. The one that dosen't "Blow Up" the cpu

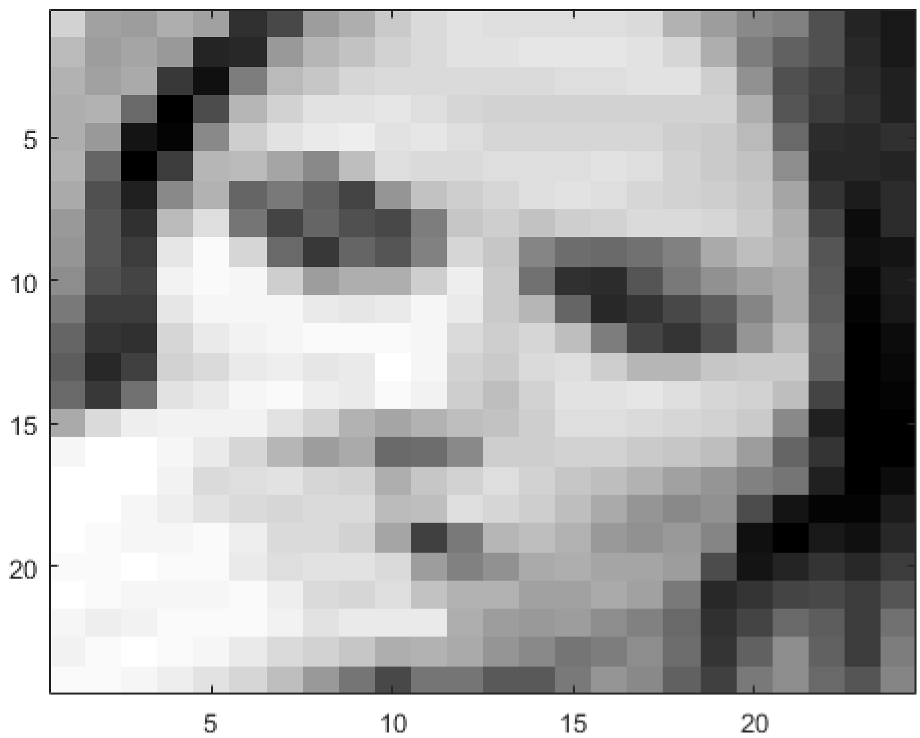
```
clc; clear all; close all;
```

Load the data that was extracted form the csv file earlier.

```
load TestingPixels.mat
load TrainingPixels.mat
```

Turn row data into a 48x48 img and resize

```
i = 3 ; % random image
figure ; colormap gray;
fim = reshape(trainingPixels(i, 2:end), [48,48])'; % row = 48 x 48
 image
imagesc(fim)                               % show the image
title(num2str(trainingPixels(i, 1)))       % show the label


sfim = imresize(fim, 0.5);                 % Resize to 24 x 24
 img
imagesc(sfim);
```



Frequency componenets from Nick submission

Applying the filters on input images

```
im1_fft  = fft2(sfim);

gh = fftshift(im1_fft);
```

Nuetralizing the Phase to display Magnitude only

```
im1_M = abs(gh);
```

Inverse fft2

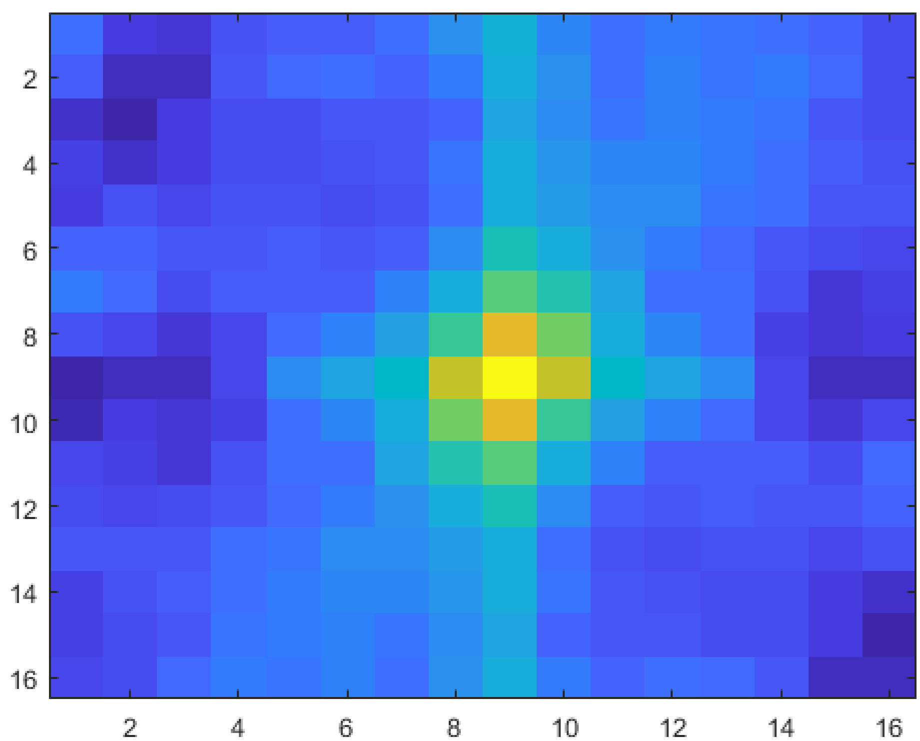
```
restoredP1 = log(abs(ifft2(im1_M*exp(1i*0)))+1);

re = fftshift(restoredP1);
```

Calculating plotting limits

```
I_Mag_min = min(min(abs(restoredP1)));
I_Mag_max = max(max(abs(restoredP1)));

figure;
imshow(abs(re),[I_Mag_min I_Mag_max ]);
```

Extract lower frequencies by just cutting to 16 x 16

```
newRe = re(5:20,5:20);
figure; imagesc(newRe);
```



Reshape to return to NN

```
stuff = reshape(newRe, [1,256]);
```

Published with MATLAB® R2017b