Matt Washburn

Kristine Nutter

Brandon Bench

Nick DeMarco

Delivery Report

When creating the hybrid image of two different images using the magnitude and phase components of the images, you must first take the fast Fourier transform of the two images. You then have to isolate the magnitude and phase of the images by using the **abs()** function (to isolate magnitude) and **angle()** function (to isolate the phase). Once you get these different components, you then have to recompute the frequency, by using the formula: **output1 = mag1 .* exp(1i*phase2)**, using the magnitude component of one image and the phase of the other. You do this for both images. You then take the inverse of the output from the above formula and that gives you the hybrid image of the two images.

When neutralizing the magnitude, we extract the phase after applying the fast Fourier transform on an image. To do this, we use **exp(1i*angle(im1_fft))** the '**angle()**' function that returns the phase angles, in radians for each element of the object. In order to restore the image with just the phase, apply the inverse fast Fourier transform after the phase angles have been calculated.

To remove the phase, first use fast Fourier transform, then you need to use the **fftshift**. Use '**abs()**' to get the complex magnitude of the image. You then need to take the inverse of this complex magnitude by using the formula: **log(abs(ifft2(im1_M*exp(1i*0)))+1)**. This will give you the restored image, which you then need to **fftshift** again. Once this is complete, you must then calculate the plotting limits to show the phase removed image. You do this by using these formulas: **I_Mag_min = min(min(abs(restoredP1))) and I_Mag_max = max(max(abs(restoredP1)))**. Once this is complete, you can use those max and mins to show the phase removed images.

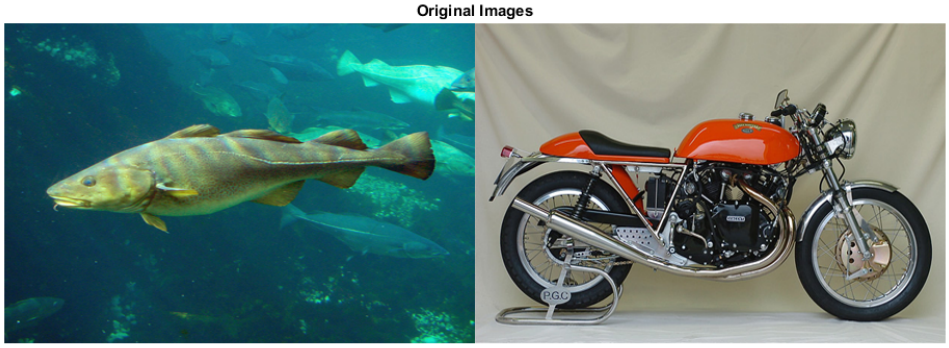
Table of Contents


```
% Author: Nick DeMarco
% Hybrid Image

clc;
clear;
close all; % closes all figures
```

Setup

```
image1 = imread('fish.bmp');
image2 = imread('motorcycle.bmp');

image1 = imresize(image1,[307 453]);
image2 = imresize(image2,[307 453]);

figure('Name', 'Original
 Images','NumberTitle','off');imshowpair(image1, image2, 'montage');
title("Original Images");

image1double = double(image1)/255;
image2double = double(image2)/255;

im1 = rgb2gray(image1double);
im2 = rgb2gray(image2double);

figure('Name', 'Grayscale Images','NumberTitle','off');imshowpair(im1,
 im2, 'montage');
title("Grayscale Images");

[im1h, im1w] = size(im1);
[im2h, im2w] = size(im2);

rows = max(im1h, im2h);
cols = max(im1w, im2w);
```

Original Images



Grayscale Images

Take the FFT of the two images

```
im1_FFT = fft2(im1, rows, cols);
im2_FFT = fft2(im2, rows, cols);
```

Find magnitude and phase of the two images

```
mag1 = abs(im1_FFT);
mag2 = abs(im2_FFT);

phase1 = angle(im1_FFT);
phase2 = angle(im2_FFT);
```

Recompute the frequency

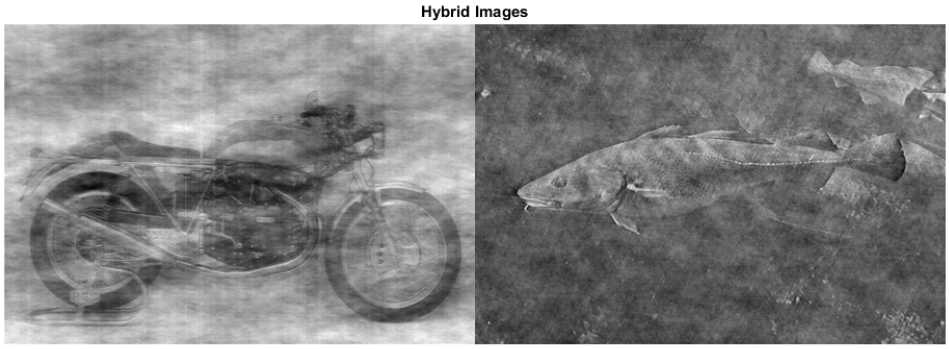
```
output1 = mag1 .* exp(1i*phase2);
output2 = mag2 .* exp(1i*phase1);
```

Find inverse images

```
inv1 = real(ifft2(output1));
inv2 = real(ifft2(output2));
```

Diplay New Hybrid Images

```
figure('Name', 'Hybrid Images','NumberTitle','off');imshowpair(inv1,
 inv2, 'montage');
title("Hybrid Images");
```



Published with MATLAB® R2017a

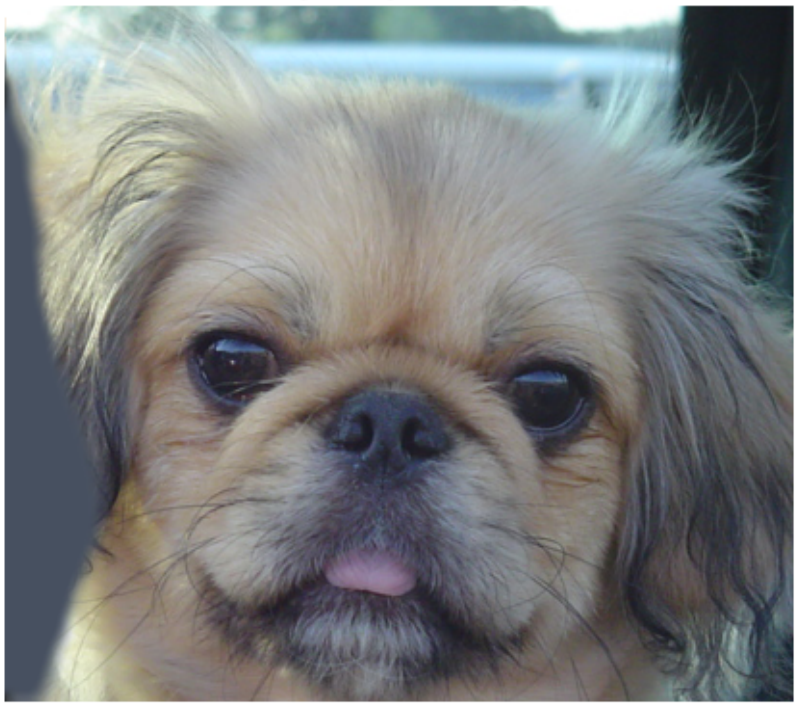
Table of Contents


```
% Author: Brandon Bench
% Hybrid Image

clc;
clear;
close all; % closes all figures
```

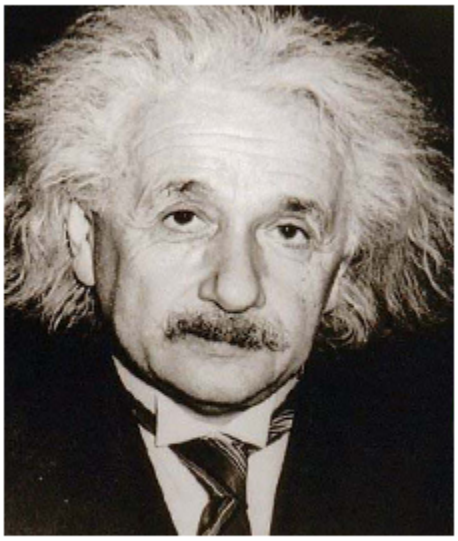
Setup

```
image1 = imread('dog.bmp');
image2 = imread('einstein.bmp');
image3 = imread('fish.bmp');

figure; imshow(image1);
title("Dog - Original Image");
figure; imshow(image2);
title("Einstein - Original Image");
figure; imshow(image3);
title("Fish - Original Image");

image1double = double(image1)/255;
image2double = double(image2)/255;
image3double = double(image3)/255;

im1 = rgb2gray(image1double);
im2 = rgb2gray(image2double);
im3 = rgb2gray(image3double);

figure; imshow(im1);
title("Dog - Grayscale Image");
figure; imshow(im2);
title("Einstein - Grayscale Image");
figure; imshow(im3);
title("Fish - Grayscale Image");
```

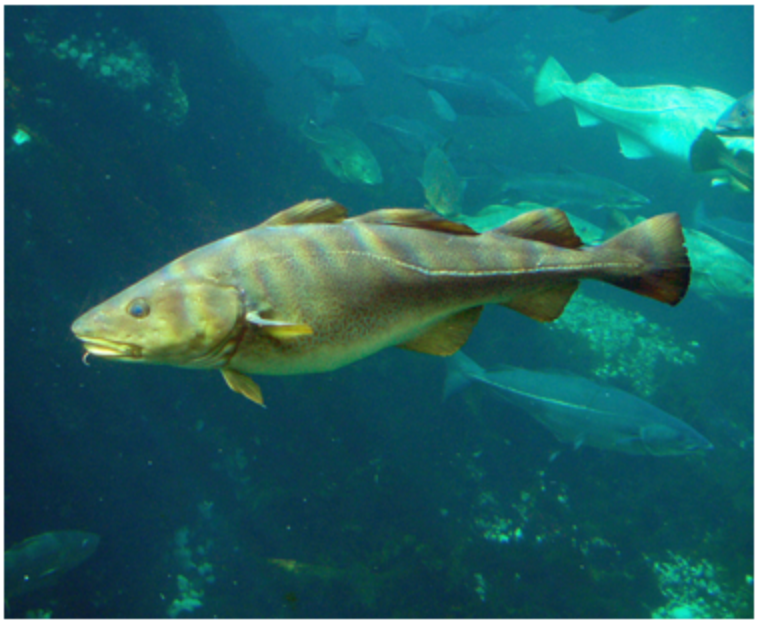
Dog - Original Image



Einstein - Original Image

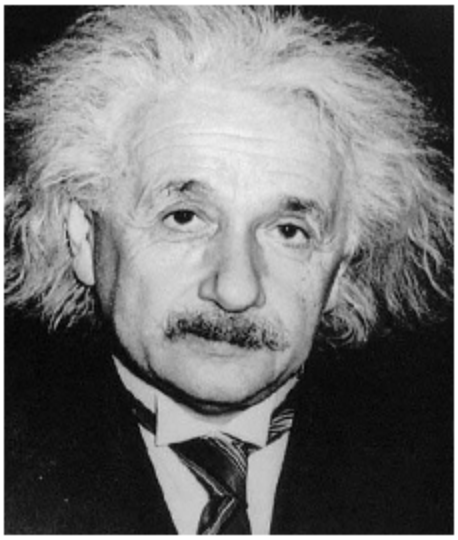


Fish - Original Image



Dog - Grayscale Image

Einstein - Grayscale Image



Fish - Grayscale Image

Applying the filters on input images

```
im1_fft  = fft2(im1);
im2_fft  = fft2(im2);
```

```
im3_fft  = fft2(im3);
```

Nuetralizing the Magnitude to display Phase only

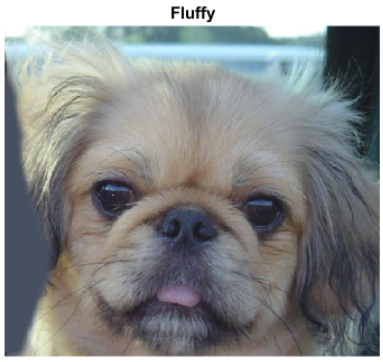
```
im1_P = exp(1i*angle(im1_fft));
im2_P = exp(1i*angle(im2_fft));
im3_P = exp(1i*angle(im3_fft));
```

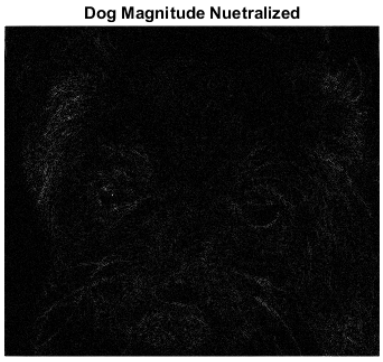
Inverse fft2

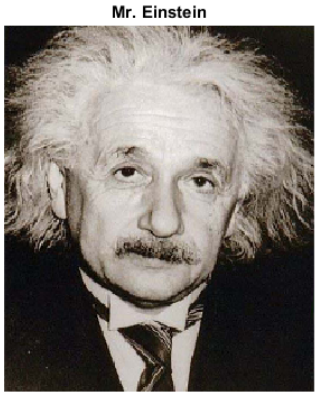
```
restoredP1 = ifft2(im1_P);
restoredP2 = ifft2(im2_P);
restoredP3 = ifft2(im3_P);
```

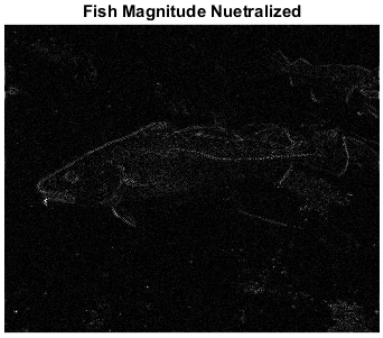
Calculating plotting limits

```
I_Phase_min = min(min(abs(restoredP1)));
I_Phase_max = max(max(abs(restoredP1)));


figure('position', [200, 200, 1000, 400]); subplot(1,2,1),
 imshow(image1), title("Fluffy")
subplot(1,2,2),
imshow(abs(restoredP1),[I_Phase_min I_Phase_max ]);
title("Dog Magnitude Nuetralized")

figure('position', [200, 200, 1000, 400]); subplot(1,2,1),
 imshow(image2), title("Mr. Einstein")
subplot(1,2,2),
imshow(abs(restoredP2),[I_Phase_min I_Phase_max ]);
title("Albert Magnitude Nuetralized")

figure('position', [200, 200, 1000, 400]); subplot(1,2,1),
 imshow(image3), title("Pescado")
subplot(1,2,2),
imshow(abs(restoredP3),[I_Phase_min I_Phase_max ]);
title("Fish Magnitude Nuetralized")
```









Published with MATLAB® R2017a

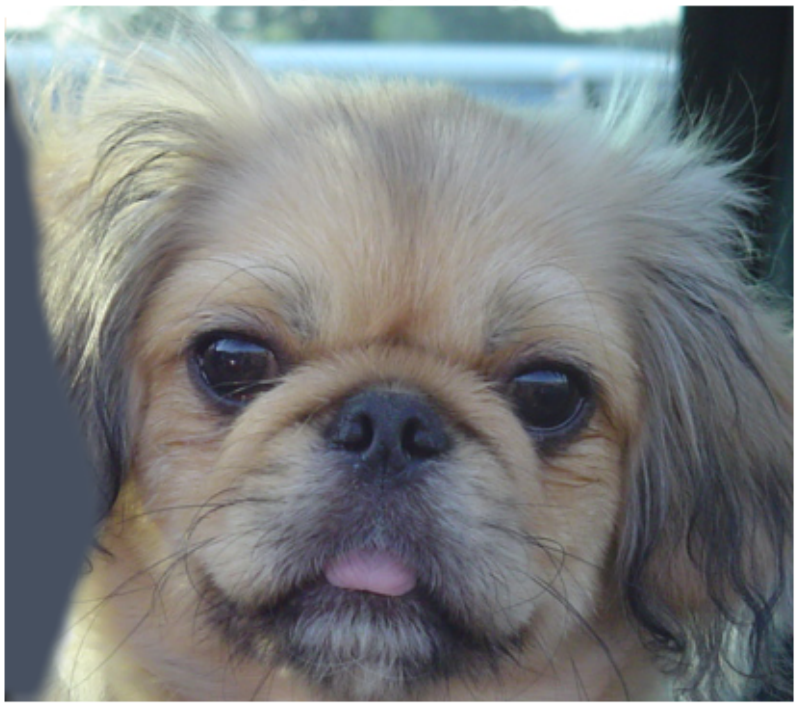
Table of Contents


```
% Author: Nick DeMarco
% Hybrid Image

clc;
clear;
close all; % closes all figures
```

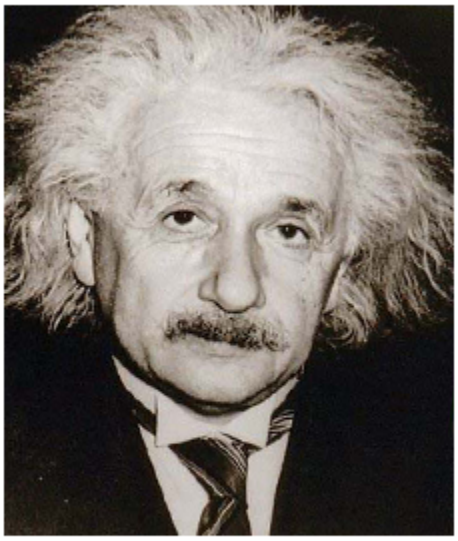
Setup

```
image1 = imread('dog.bmp');
image2 = imread('einstein.bmp');
image3 = imread('fish.bmp');

figure; imshow(image1);
title("Dog - Original Image");
figure; imshow(image2);
title("Einstein - Original Image");
figure; imshow(image3);
title("Fish - Original Image");

image1double = double(image1)/255;
image2double = double(image2)/255;
image3double = double(image3)/255;

im1 = rgb2gray(image1double);
im2 = rgb2gray(image2double);
im3 = rgb2gray(image3double);

figure; imshow(im1);
title("Dog - Grayscale Image");
figure; imshow(im2);
title("Einstein - Grayscale Image");
figure; imshow(im3);
title("Fish - Grayscale Image");
```

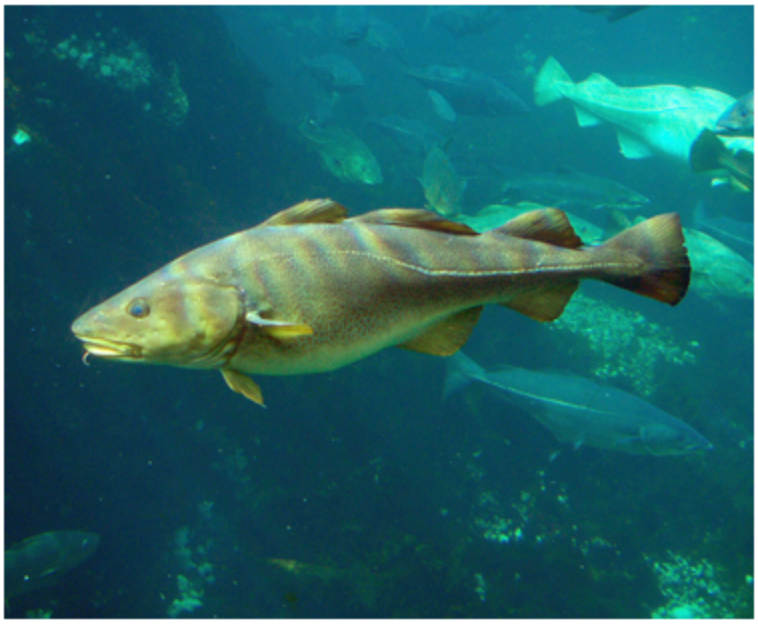
Dog - Original Image



Einstein - Original Image

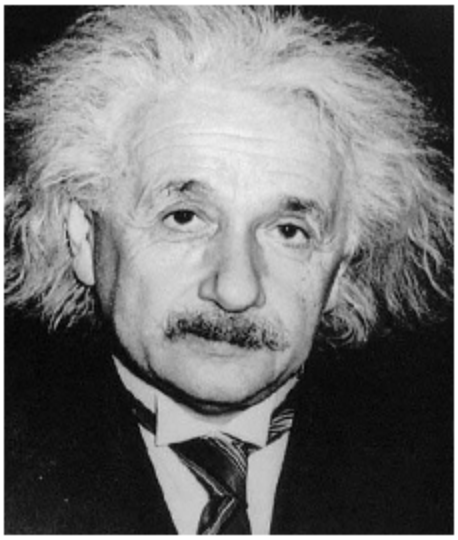


Fish - Original Image



Dog - Grayscale Image

Einstein - Grayscale Image



Fish - Grayscale Image

Applying the filters on input images

```
im1_fft  = fft2(im1);
im2_fft  = fft2(im2);
```

```
im3_fft  = fft2(im3);

gh = fftshift(im1_fft);
g2 = fftshift(im2_fft);
g3 = fftshift(im3_fft);
```

Nuetralizing the Phase to display Magnitude only

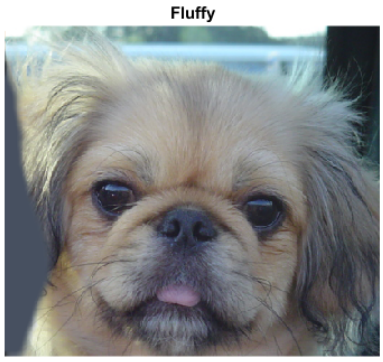
```
im1_M = abs(gh);
im2_M = abs(g2);
im3_M = abs(g3);
```

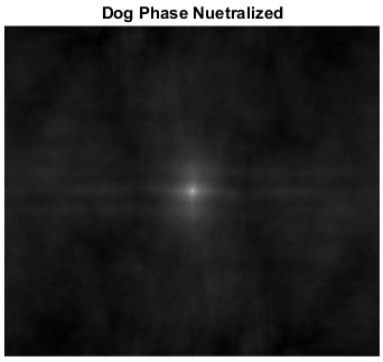
Inverse fft2

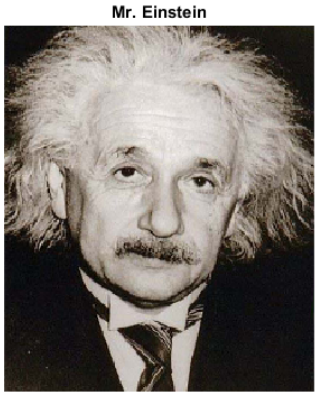
```
restoredP1 = log(abs(ifft2(im1_M*exp(1i*0)))+1);
restoredP2 = log(abs(ifft2(im2_M*exp(1i*0)))+1);
restoredP3 = log(abs(ifft2(im3_M*exp(1i*0)))+1);

re = fftshift(restoredP1);
r1 = fftshift(restoredP2);
r2 = fftshift(restoredP3);
```

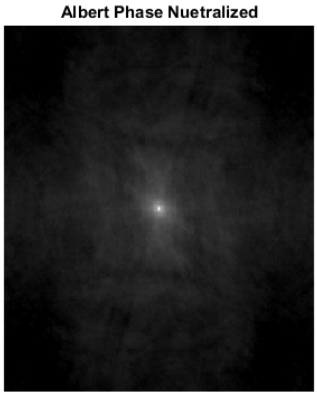
Calculating plotting limits

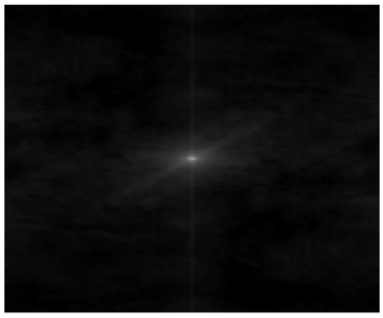
```
I_Mag_min = min(min(abs(restoredP1)));
I_Mag_max = max(max(abs(restoredP1)));


figure('position', [200, 200, 1000, 400]); subplot(1,2,1),
 imshow(image1), title("Fluffy")
subplot(1,2,2),
imshow(abs(re),[I_Mag_min I_Mag_max ]);
title("Dog Phase Nuetralized")

figure('position', [200, 200, 1000, 400]); subplot(1,2,1),
 imshow(image2), title("Mr. Einstein")
subplot(1,2,2),
imshow(abs(r1),[I_Mag_min I_Mag_max ]);
title("Albert Phase Nuetralized")

figure('position', [200, 200, 1000, 400]); subplot(1,2,1),
 imshow(image3), title("Pescado")
subplot(1,2,2),
imshow(abs(r2),[I_Mag_min I_Mag_max ]);
title("Fish Phase Nuetralized")
```











Published with MATLAB® R2017a

HW 1.C for 1.a.i (Hybrid Image)

Tic Toc Analysis

Critical Block	**Elapsed Time**
Reading in Images	0.159088 seconds
Image Resize	0.134052 seconds
Displaying Original Images	2.495655 seconds
Converting Images to Double	0.002885 seconds
Images to Grayscale	0.015694 seconds
Displaying Grayscale Images	0.263186 seconds
Finding size of images	0.000242 seconds
Initializing Rows and Cols	0.000188 seconds
Taking FFT	0.039679 seconds
Finding Magnitude	0.003576 seconds
Finding Phase	0.013051 seconds
Recomputing Frequency	0.008191 seconds
Taking Inverse	0.033244 seconds
Displaying Hybrid Images	0.254677 seconds

In the first part, FFT took around .003 seconds to run.

Profile Summary

Generated 02-Mar-2018 11:11:38 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
HW1_Hybrid	1	0.134 s	0.016 s	
imread	2	0.051 s	0.006 s	
imread>call_format_specific_reader	2	0.039 s	0.001 s	
imagesci\private\readbmp	2	0.037 s	0.001 s	
imresize	2	0.035 s	0.007 s	
imagesci\private\imbmpinfo	2	0.029 s	0.002 s	
...\imbmpinfo>initializeBMPInfoStruct	2	0.019 s	0.001 s	
ima...i\private\initializeMetadataStruct	2	0.018 s	0.002 s	
datestr	2	0.016 s	0.002 s	
timefun\private\dateformverify	2	0.013 s	0.001 s	
timefun\private\formatdate	2	0.012 s	0.007 s	
imresize>parseInputs	2	0.011 s	0.003 s	
fft2	2	0.010 s	0.010 s	
ifft2	2	0.008 s	0.008 s	
imagesci\private\readbmpdata	2	0.008 s	0.001 s	
close	1	0.007 s	0.002 s	
ima...ivate\readbmpdata>bmpReadData24	2	0.007 s	0.004 s	
imresize>resizeAlongDim	4	0.007 s	0.003 s	
imagesci\private\imbmpinfo>readBMPInfo	2	0.006 s	0.001 s	
cnv2icudf	2	0.005 s	0.005 s	
contributions	4	0.005 s	0.004 s	
imagesci\private\imbmpinfo>readWin3xInfo	2	0.005 s	0.001 s	
close>safegetchildren	1	0.004 s	0.000 s	

angle	2	0.004 s	0.004 s	
allchild	1	0.004 s	0.002 s	
imresize>fixupSizeAndScale	2	0.003 s	0.002 s	
fileparts	2	0.003 s	0.002 s	
stringToChar	2	0.002 s	0.001 s	
imresize>parsePreMethodArgs	2	0.002 s	0.002 s	
rgb2gray	2	0.002 s	0.002 s	
...te\imbmpinfo>readWin3xBitmapHeader	2	0.002 s	0.002 s	
ima...rivate\readbmpdata>readFromFile	2	0.002 s	0.002 s	
imagesci\private\getFileFromURL	2	0.002 s	0.001 s	
resizeAlongDimUsingNearestNeighbor	2	0.002 s	0.002 s	
imresizemex (MEX-file)	2	0.002 s	0.002 s	
onCleanup>onCleanup.delete	5	0.002 s	0.001 s	
contains	2	0.001 s	0.001 s	
rot90	2	0.001 s	0.001 s	
stringToChar>@(x)convertToChar(x)	4	0.001 s	0.001 s	
cubic	4	0.001 s	0.001 s	
imread>get_full_filename	2	0.001 s	0.001 s	
cell.ismember	2	0.001 s	0.001 s	
isPureNearestNeighborComputation	8	0.001 s	0.001 s	
findFirstParamString	2	0.001 s	0.001 s	
deriveScaleFromSize	2	0.001 s	0.001 s	
imresize>findMethodArg	2	0.001 s	0.001 s	
rgb2gray>parse_inputs	2	0.001 s	0.001 s	
imresize>fixupSize	2	0.001 s	0.001 s	
imresize>scaleOrSize	2	0.001 s	0.001 s	
imagesci\private\imbmpinfo>getSignature	2	0.001 s	0.001 s	

ima...ate\imbmpinfo>readBMPFileHeader	2	0.001 s	0.001 s	
imresize>postprocessImage	2	0.001 s	0.001 s	
imagesci\private\imbmpinfo>postProcess	2	0.001 s	0.001 s	
imread>parse_inputs	2	0.001 s	0.001 s	
imresize>checkForMissingOutputArgument	2	0.001 s	0.000 s	
...te\imbmpinfo>readVersion3xColormap	2	0.001 s	0.001 s	
imresize>preprocessImage	2	0.000 s	0.000 s	
imresize>warnIfPostMethodArgs	2	0.000 s	0.000 s	
stringToChar>convertToChar	4	0.000 s	0.000 s	
close>getEmptyHandleList	1	0.000 s	0.000 s	
onCleanup>onCleanup.onCleanup	5	0.000 s	0.000 s	
strfun\private\isTextStrict	2	0.000 s	0.000 s	
...set(rootobj,'ShowHiddenHandles',Temp)	1	0.000 s	0.000 s	
allchild>getchildren	1	0.000 s	0.000 s	
close>request_close	1	0.000 s	0.000 s	
dimensionOrder	2	0.000 s	0.000 s	
imresize>parseParamValuePairs	2	0.000 s	0.000 s	
stringToLegacyText	2	0.000 s	0.000 s	
close>checkfigs	1	0.000 s	0.000 s	
datestr>getdateform	2	0.000 s	0.000 s	
ima...rivate\imbmpinfo>@()fclose(fid)	2	0.000 s	0.000 s	
uitools\private\allchildRootHelper	1	0.000 s	0.000 s	
ima...ate\imbmpinfo>decodeCompression	2	0.000 s	0.000 s	
imresize>isInputIndexed	6	0.000 s	0.000 s	
ima...vate\readbmpdata>@()fclose(fid)	2	0.000 s	0.000 s	
ispc	2	0.000 s	0.000 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self

time also includes overhead resulting from the process of profiling.

HW 1.C for 1.a.ii (Magnitude Neutralization)

1.c) We believe that the Fast Fourier Transform will be the most time consuming machine learning algorithm. It is the main algorithm we are using in this section of the homework.

1.c.i) This table is based on one function call per image. Not a combined time for all three images.
Also to note, these runtimes can vary widely each time the program is run.

Code Block	**Elapsed Time (averaged over 4 runs)**
Imread image	0.585700 E-02 seconds
Convert to Double type	0.326500 E-02 seconds
Convert to Grayscale	0.183425 E-02 seconds
2-D FFT	1.012050 E-02 seconds
Neutralizing the Magnitude	1.249530 E-02 seconds
2-D Inverse FFT	1.309500 E-02 seconds
Calculate Plot Limits to graph	1.752750 E-02 seconds

1.c.ii.2) fft2 and ifft2 each took 0.009 s and 0.006 s to run. Based off of one run.

Profile Summary

Generated 01-Mar-2018 17:11:12 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
HW1_1_a_ii	1	0.104 s	0.016 s	
imread	3	0.054 s	0.007 s	
imread>call_format_specific_reader	3	0.042 s	0.001 s	
imagesci\private\readbmp	3	0.040 s	0.001 s	
imagesci\private\imbmpinfo	3	0.030 s	0.003 s	
...\imbmpinfo>initializeBMPInfoStruct	3	0.019 s	0.001 s	
ima...i\private\initializeMetadataStruct	3	0.018 s	0.002 s	
datestr	3	0.016 s	0.002 s	
timefun\private\dateformverify	3	0.014 s	0.001 s	
timefun\private\formatdate	3	0.012 s	0.007 s	
close	1	0.011 s	0.003 s	
ifft2	3	0.009 s	0.009 s	
imagesci\private\readbmpdata	3	0.009 s	0.001 s	
ima...ivate\readbmpdata>bmpReadData24	3	0.008 s	0.004 s	
close>safegetchildren	1	0.007 s	0.001 s	
imagesci\private\imbmpinfo>readBMPInfo	3	0.007 s	0.001 s	
fft2	3	0.006 s	0.006 s	
allchild	1	0.006 s	0.003 s	
imagesci\private\imbmpinfo>readWin3xInfo	3	0.006 s	0.002 s	
cnv2icudf	3	0.005 s	0.005 s	
angle	3	0.004 s	0.004 s	
rgb2gray	3	0.003 s	0.002 s	
...te\imbmpinfo>readWin3xBitmapHeader	3	0.003 s	0.002 s	
ima...rivate\readbmpdata>readFromFile	3	0.003 s	0.002 s	
fileparts	3	0.003 s	0.002 s	

onCleanup>onCleanup.delete	7	0.002 s	0.001 s	
rot90	3	0.002 s	0.002 s	
imread>get_full_filename	3	0.001 s	0.001 s	
imread>parse_inputs	3	0.001 s	0.001 s	
cell.ismember	3	0.001 s	0.001 s	
ima...ate\imbmpinfo>readBMPFileHeader	3	0.001 s	0.001 s	
imagesci\private\imbmpinfo>getSignature	3	0.001 s	0.001 s	
rgb2gray>parse_inputs	3	0.001 s	0.001 s	
imagesci\private\imbmpinfo>postProcess	3	0.001 s	0.001 s	
...set(rootobj,'ShowHiddenHandles',Temp)	1	0.001 s	0.001 s	
...te\imbmpinfo>readVersion3xColormap	3	0.001 s	0.001 s	
allchild>getchildren	1	0.001 s	0.001 s	
imagesci\private\getFileFromURL	3	0.001 s	0.001 s	
onCleanup>onCleanup.onCleanup	7	0.001 s	0.001 s	
close>request_close	1	0.000 s	0.000 s	
uitools\private\allchildRootHelper	1	0.000 s	0.000 s	
close>getEmptyHandleList	1	0.000 s	0.000 s	
close>checkfigs	1	0.000 s	0.000 s	
ima...vate\readbmpdata>@()fclose(fid)	3	0.000 s	0.000 s	
datestr>getdateform	3	0.000 s	0.000 s	
ispc	3	0.000 s	0.000 s	
ima...ate\imbmpinfo>decodeCompression	3	0.000 s	0.000 s	
ima...rivate\imbmpinfo>@()fclose(fid)	3	0.000 s	0.000 s	

HW 1.C for 1.a.iii (Phase Removal)

Tic Toc Analysis

Critical Block	**Elapsed Time**
Reading in Images	0.024449 seconds
Displaying Original Images	0.490118 seconds
Converting Images to Double	0.003109 seconds
Images to Grayscale	0.002685 seconds
Displaying Grayscale Images	0.433551 seconds
FFT2	0.010948 seconds
FFTShift	0.012342 seconds
Neutralizing Phase	0.003293 seconds
Inverse IFFT2	0.019009 seconds
FFTShift	0.001690 seconds
Calculating Plotting Limits	0.002852 seconds
Displaying Phase Neutralized Images	0.400440 seconds

In the third part, FFT took around .012 seconds to run while the inverse function IFFT2 took around .001 seconds to run.

Profile Summary

Generated 02-Mar-2018 11:21:00 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
HW1_1_a_iii	1	0.357 s	0.016 s	
close	1	0.269 s	0.003 s	
close>request_close	1	0.239 s	0.010 s	
closereq	9	0.214 s	0.198 s	
imread	3	0.055 s	0.006 s	
imread>call_format_specific_reader	3	0.041 s	0.001 s	
imagesci\private\readbmp	3	0.040 s	0.001 s	
imagesci\private\imbmpinfo	3	0.030 s	0.003 s	
close>safegetchildren	1	0.026 s	0.004 s	
ismember	11	0.020 s	0.002 s	
...\imbmpinfo>initializeBMPInfoStruct	3	0.020 s	0.001 s	
ima...i\private\initializeMetadataStruct	3	0.019 s	0.002 s	
ismember>ismemberR2012a	11	0.019 s	0.004 s	
setdiff	2	0.018 s	0.001 s	
setdiff>setdiffR2012a	2	0.018 s	0.002 s	

datestr	3	0.017 s	0.002 s	
ismember>ismemberClassTypes	11	0.015 s	0.005 s	
close>request_close_helper	18	0.015 s	0.008 s	
timefun\private\dateformverify	3	0.014 s	0.001 s	
timefun\private\formatdate	3	0.013 s	0.007 s	
....graphics.internal.axesDestroyed(o,e)	6	0.012 s	0.011 s	
imagesci\private\readbmpdata	3	0.009 s	0.001 s	
unique	6	0.009 s	0.004 s	
ima...ivate\readbmpdata>bmpReadData24	3	0.009 s	0.004 s	
ifft2	3	0.006 s	0.006 s	
imagesci\private\imbmpinfo>readBMPInfo	3	0.006 s	0.001 s	
fft2	3	0.006 s	0.006 s	
cnv2icudf	3	0.006 s	0.006 s	
unique>uniqueR2012a	6	0.005 s	0.005 s	
imagesci\private\imbmpinfo>readWin3xInfo	3	0.005 s	0.001 s	
allchild	1	0.004 s	0.002 s	
fileparts	3	0.003 s	0.002 s	

ima...rivate\readbmpdata>readFromFile	3	0.003 s	0.002 s	
fftshift	6	0.002 s	0.002 s	
...te\imbmpinfo>readWin3xBitmapHeader	3	0.002 s	0.002 s	
rgb2gray	3	0.002 s	0.002 s	
gcbf	18	0.002 s	0.002 s	
...ddenHandles',oldUDDShowHiddenHandles)	9	0.002 s	0.002 s	
...ger>SubplotListenersManager.delete	3	0.002 s	0.002 s	
imagesci\private\getFileFromURL	3	0.002 s	0.001 s	
rot90	3	0.002 s	0.002 s	
axesDestroyed	6	0.002 s	0.002 s	
imread>get_full_filename	3	0.001 s	0.001 s	
onCleanup>onCleanup.delete	7	0.001 s	0.000 s	
contains	3	0.001 s	0.001 s	
close>checkfigs	10	0.001 s	0.001 s	
cell.ismember	3	0.001 s	0.001 s	
imagesci\private\imbmpinfo>postProcess	3	0.001 s	0.001 s	
ismember>ismemberBuiltinTypes	2	0.001 s	0.001 s	

close>getEmptyHandleList	3	0.001 s	0.001 s	
imagesci\private\imbmpinfo>getSignature	3	0.001 s	0.001 s	
imread>parse_inputs	3	0.001 s	0.001 s	
rgb2gray>parse_inputs	3	0.001 s	0.001 s	
ima...ate\imbmpinfo>readBMPFileHeader	3	0.001 s	0.001 s	
...te\imbmpinfo>readVersion3xColormap	3	0.000 s	0.000 s	
allchild>getchildren	1	0.000 s	0.000 s	
...set(rootobj,'ShowHiddenHandles',Temp)	1	0.000 s	0.000 s	
onCleanup>onCleanup.onCleanup	7	0.000 s	0.000 s	
strfun\private\isTextStrict	3	0.000 s	0.000 s	
stringToLegacyText	3	0.000 s	0.000 s	
datestr>getdateform	3	0.000 s	0.000 s	
uitools\private\allchildRootHelper	1	0.000 s	0.000 s	
ima...vate\readbmpdata>@()fclose(fid)	3	0.000 s	0.000 s	
ima...rivate\imbmpinfo>@()fclose(fid)	3	0.000 s	0.000 s	
ima...ate\imbmpinfo>decodeCompression	3	0.000 s	0.000 s	
ispc	3	0.000 s	0.000 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self

time also includes overhead resulting from the process of profiling.

When thinking about what could make our critical portions of our code run faster, we were thinking that it might be best to already have the images loaded ahead of time. We noticed that a bulk of our runtime involves loading the images. If we could eliminate this step, then we could drastically cut down on our critical portion runtime, thus improving the program's performance.

Contents


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

Delivery report Pt 2

This report contains the awnsers to questions posed in the Delivarble

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.
```

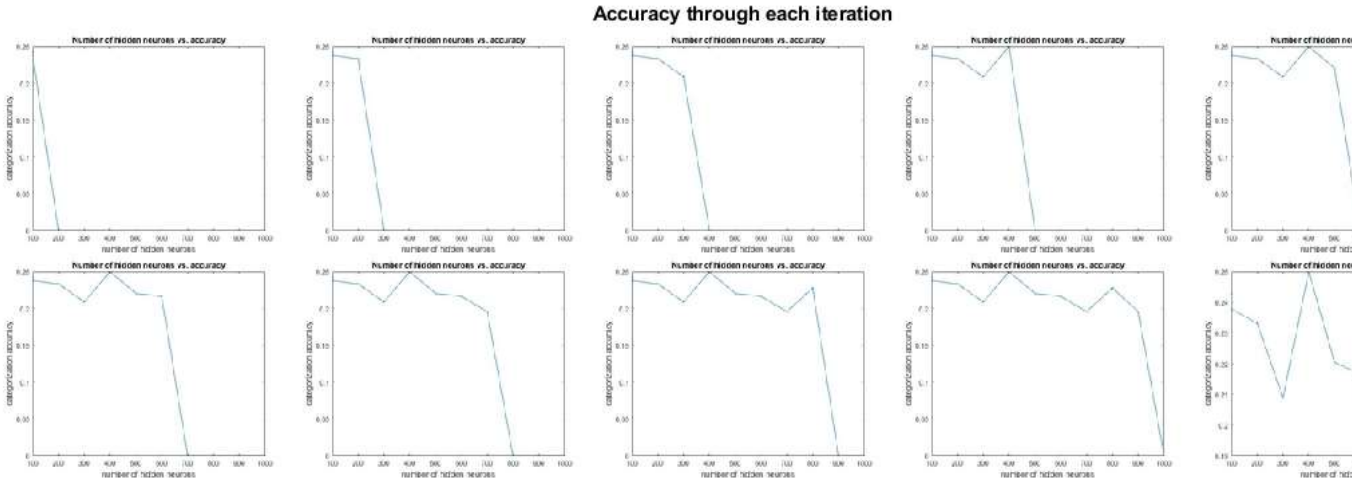
Deliverable 3cii

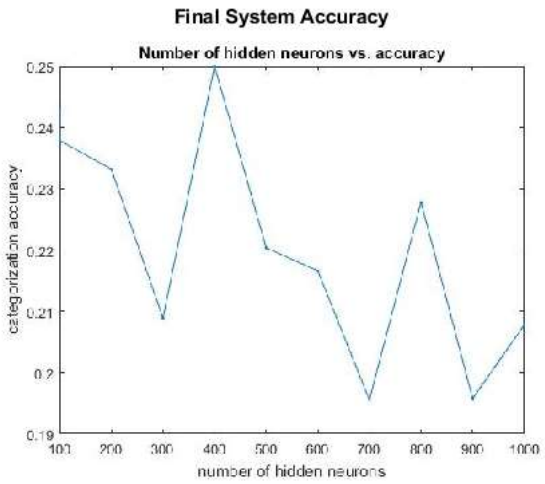
```
%The extraction method that worked better was the one with less inputs and
%more information. The extraction of the frequency magnatude components
%showed an increased accuraccy at different levels of neurons in the sweep.
%This was because the features that were fed to the input had more
%information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

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

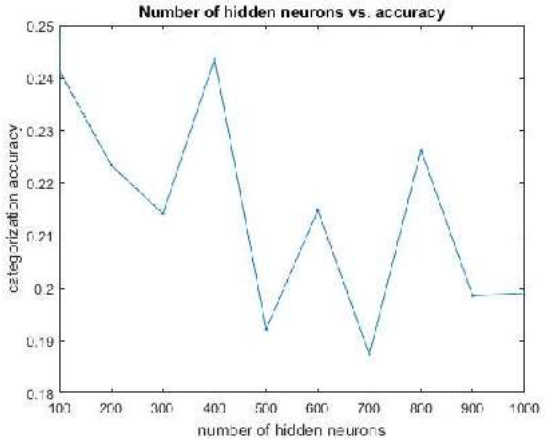
```
%Let it be known that the whole training dataset was not used in training
%this neural network. It was modified to only take 15,000 training samples
%and use 9,000 of it to sweep for the number of hidden neurons.

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 image3 image4 image5; image6 image7 image8 image9 image10];
figure; imshow(plot); title('Accuracy through each iteration');
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.

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');
figure; imshow(image1_n);
figure; imshow(image2_n);
```

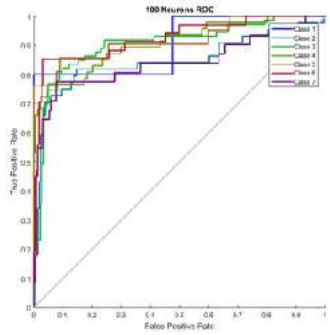
```
figure; imshow(image3_n);
figure; imshow(image4_n);
figure; imshow(image5_n);
figure; imshow(image6_n);
figure; imshow(image7_n);
figure; imshow(image8_n);
figure; imshow(image9_n);
figure; imshow(image10_n);
figure; imshow(final_n);

%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.
```

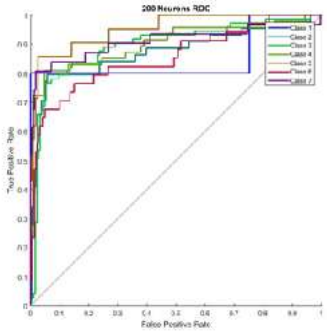


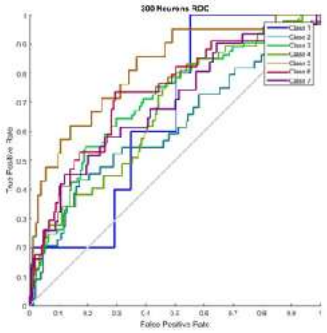


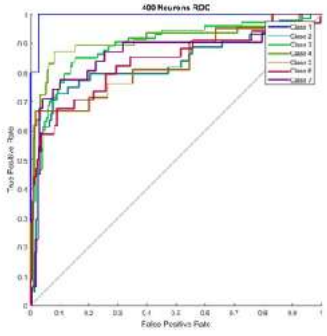
Accuracy after percentage change

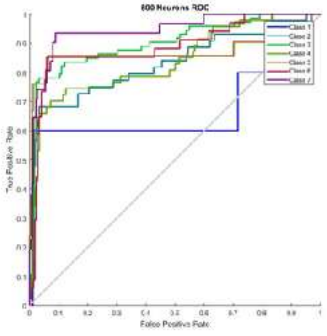


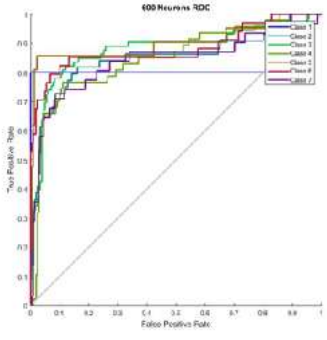


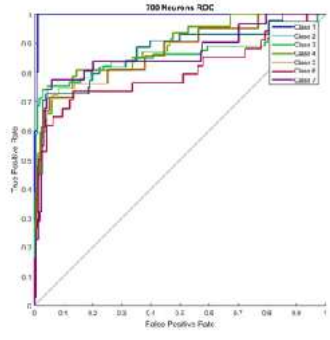


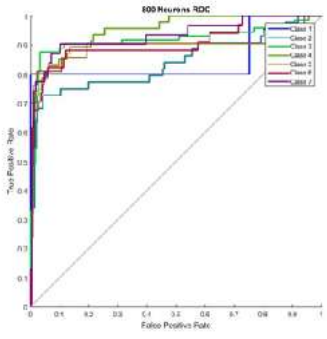


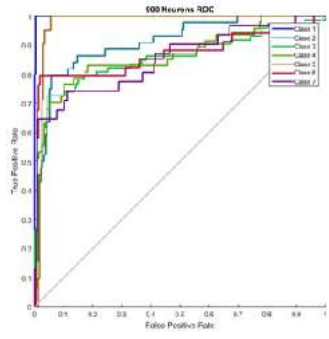


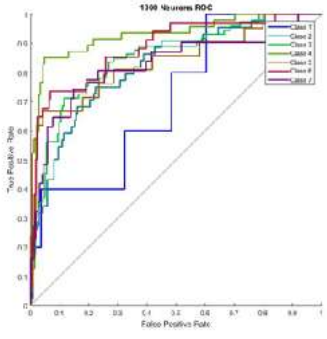


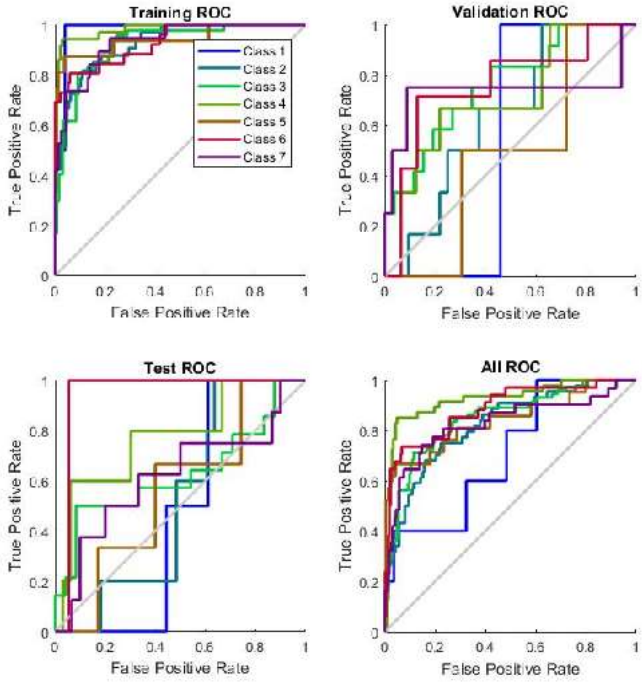

600 Neurons ROC
True Positive Rate
False Positive Rate


700 Neurons ROC
True Positive Rate
False Positive Rate


800 Neurons ROC
True Positive Rate
False Positive Rate


900 Neurons ROC
True Positive Rate
False Positive Rate


1300 Neurons ROC
True Positive Rate
False Positive Rate

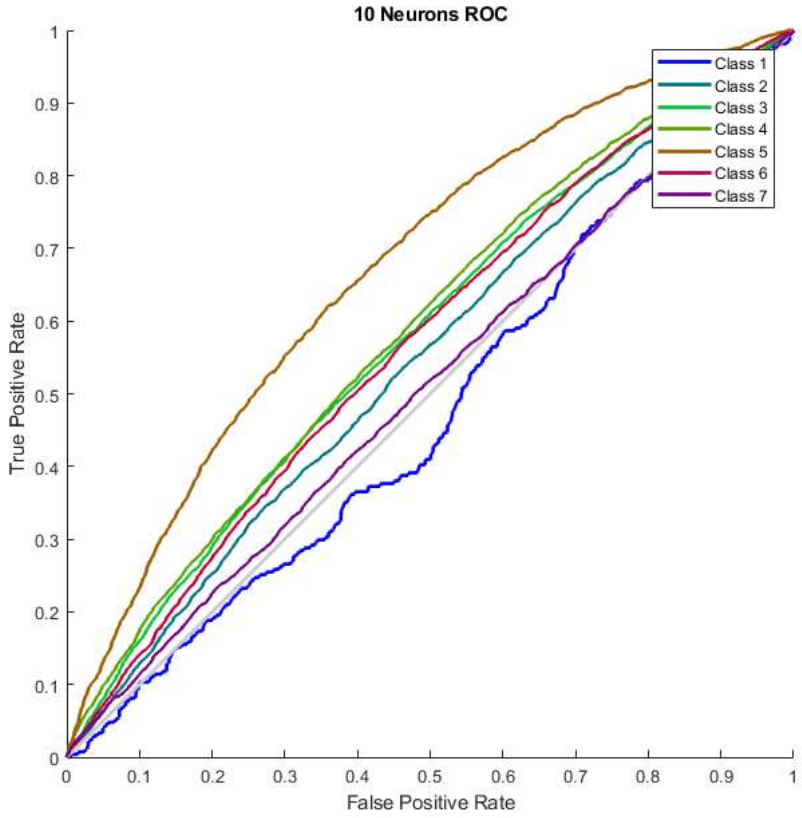


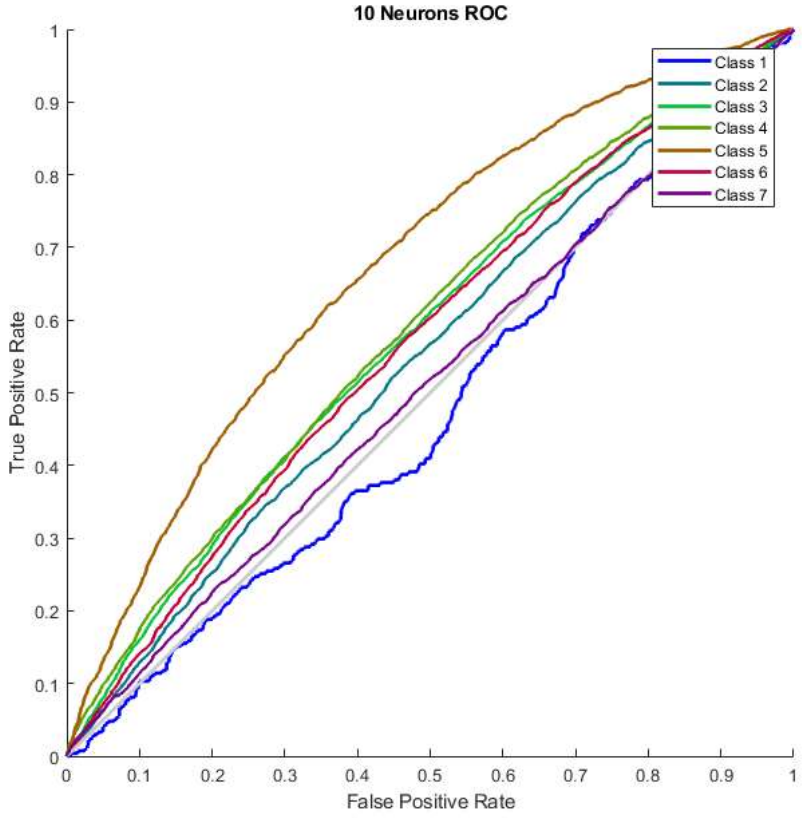
Part 4 Methods to reduce complexity

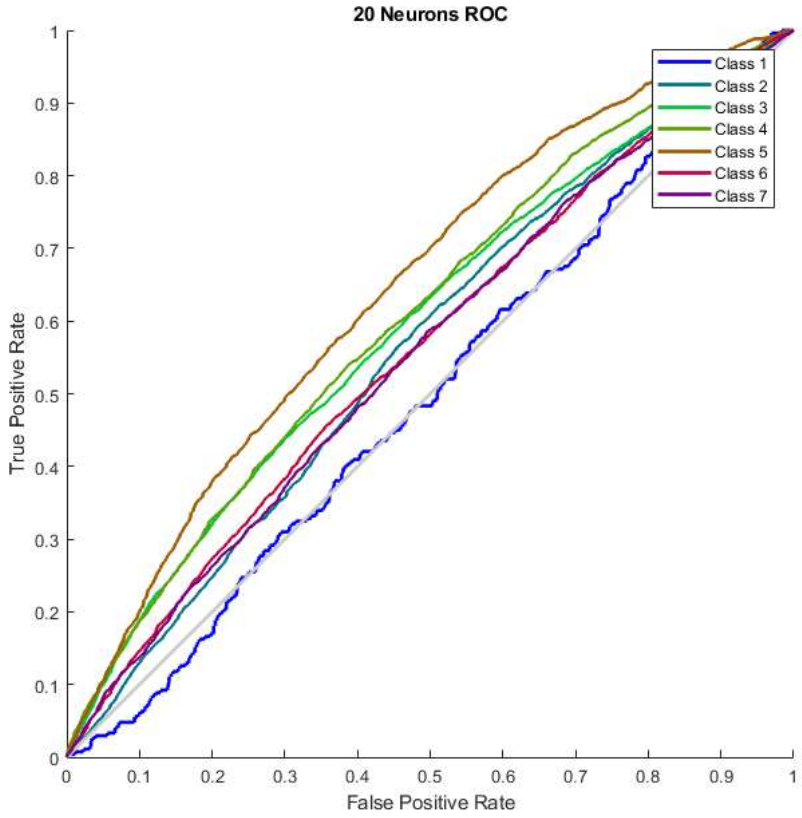
```
%Three different methods that could be used to reduce the complexity of the
%system are:
%1. wavelet: using wavelet transform would take out a lot of the noise in
%the images, allowing the neural network to grab better pixel values from
%the data sets.
%2. frequency domain: by taking the frequency 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.
```

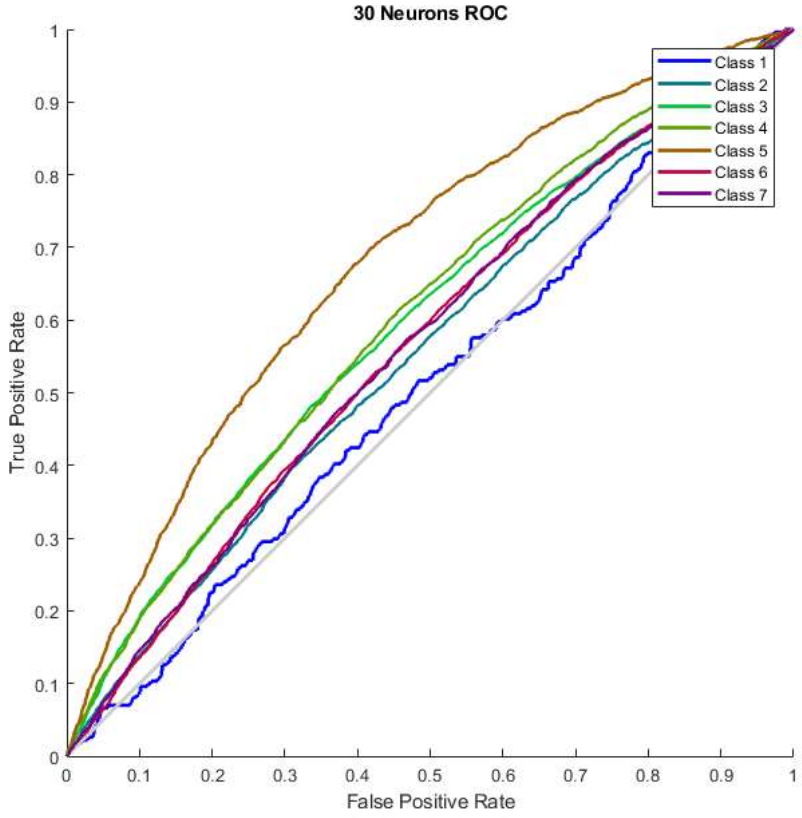
Part 5 Sweep ROC

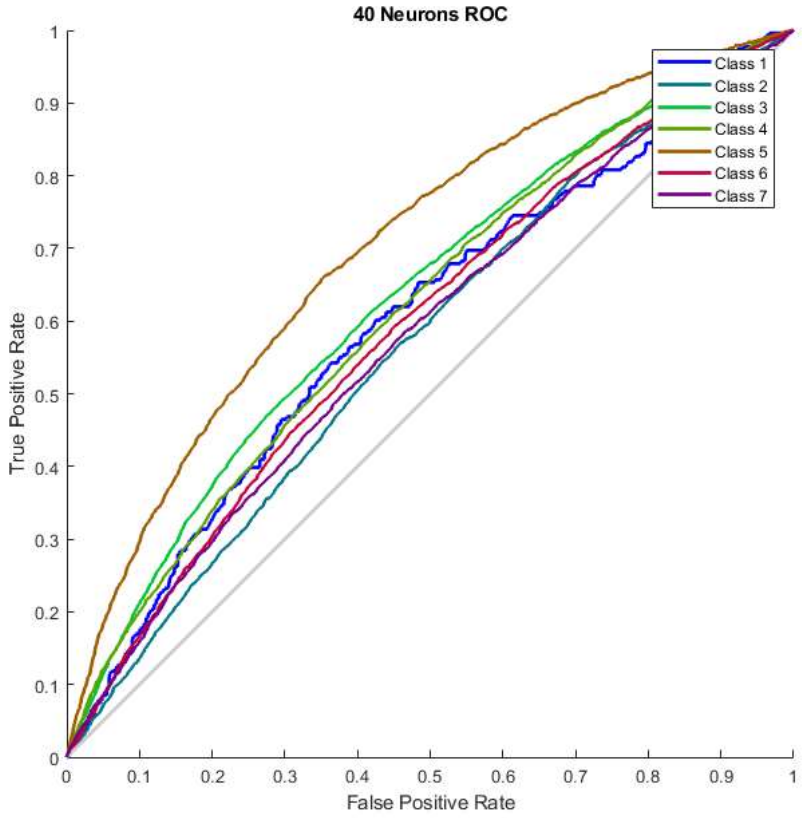
```
sweep = [10,10:10:250];

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

end
% close all
```

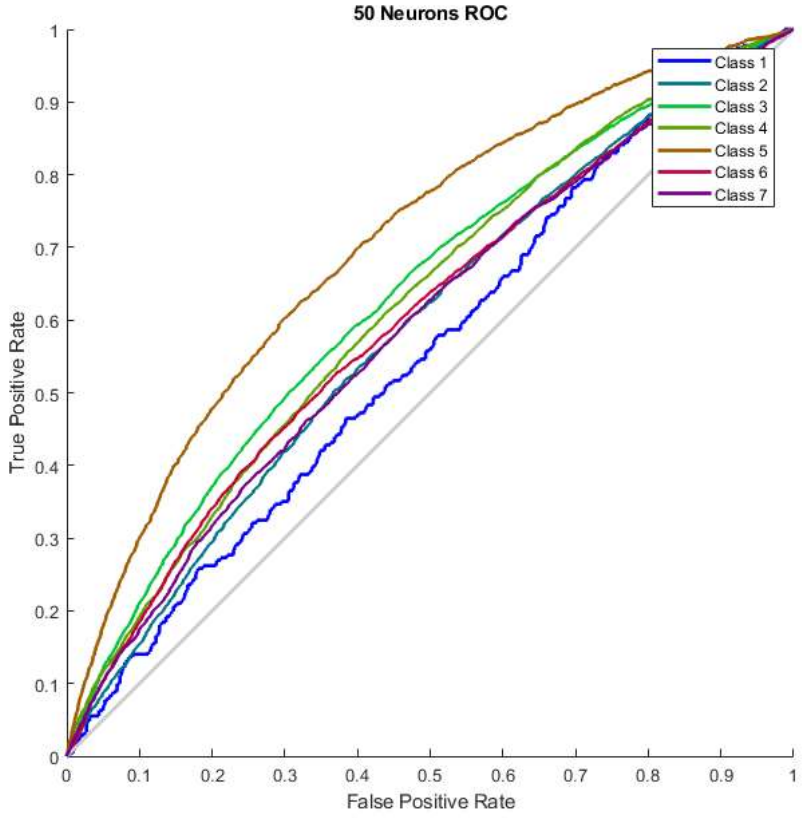

10 Neurons ROC
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
True Positive Rate
False Positive Rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

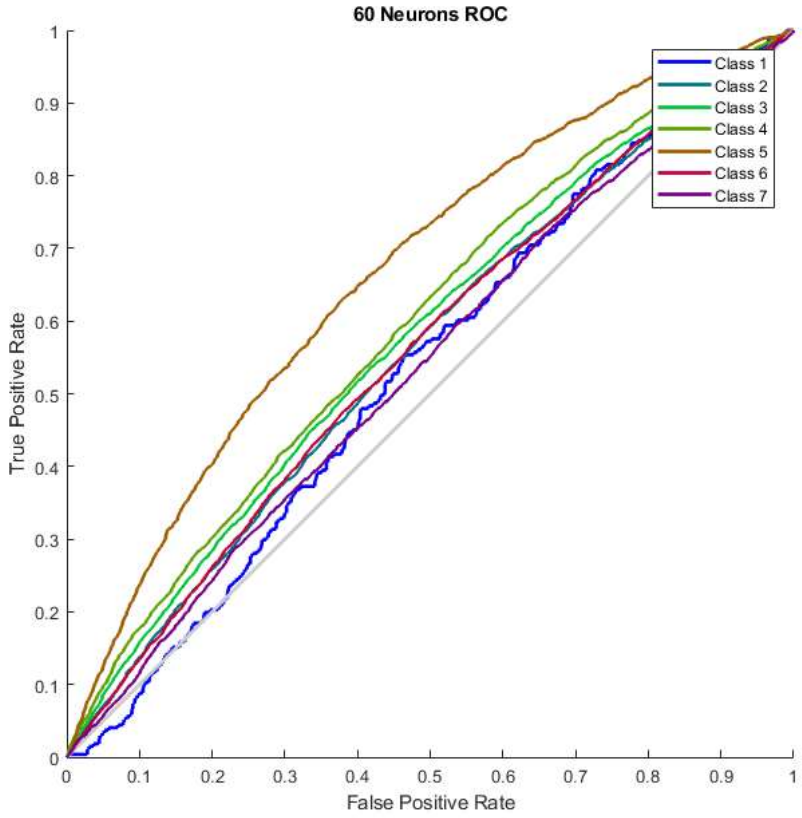

10 Neurons ROC
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
True Positive Rate
False Positive Rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

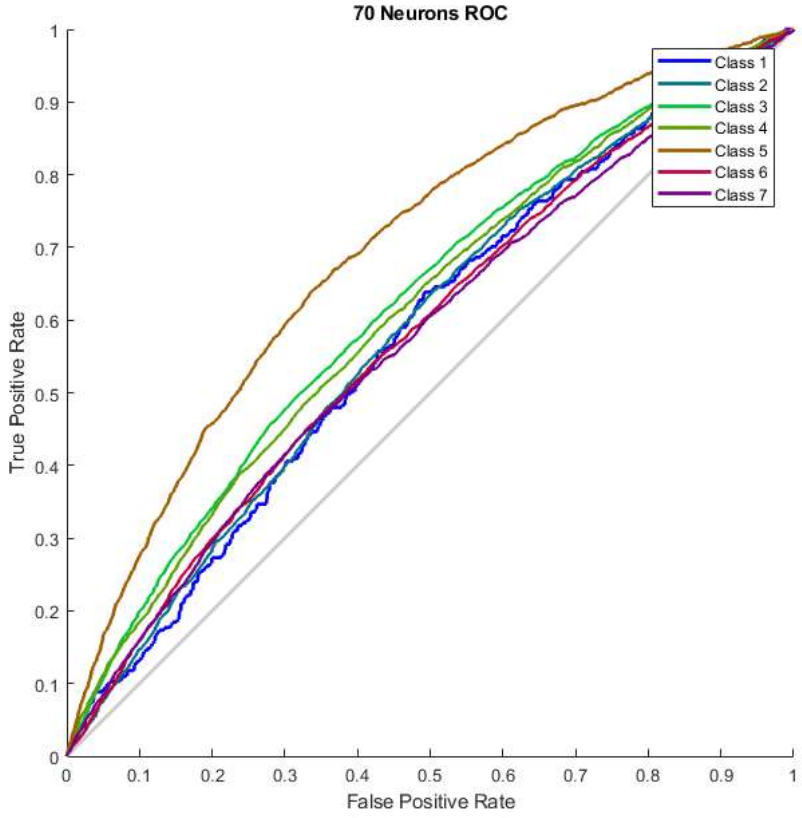

20 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

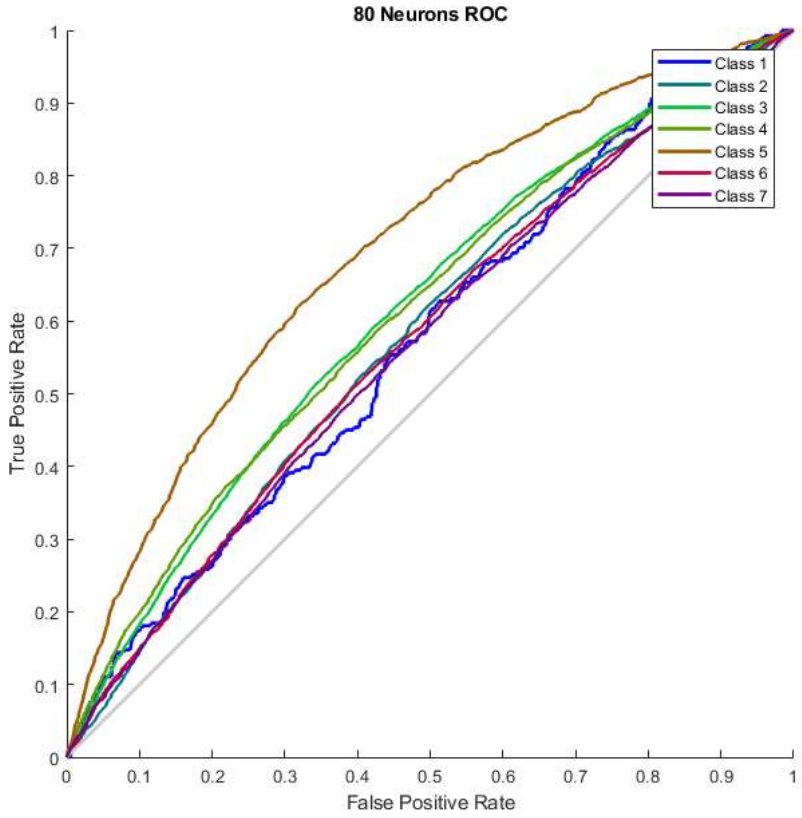

30 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

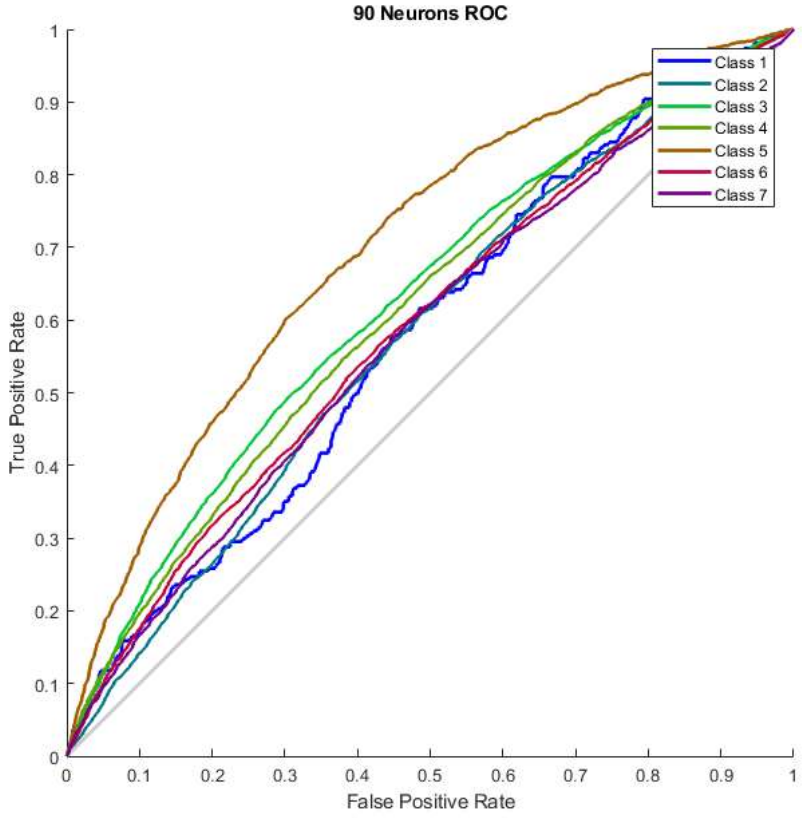

40 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

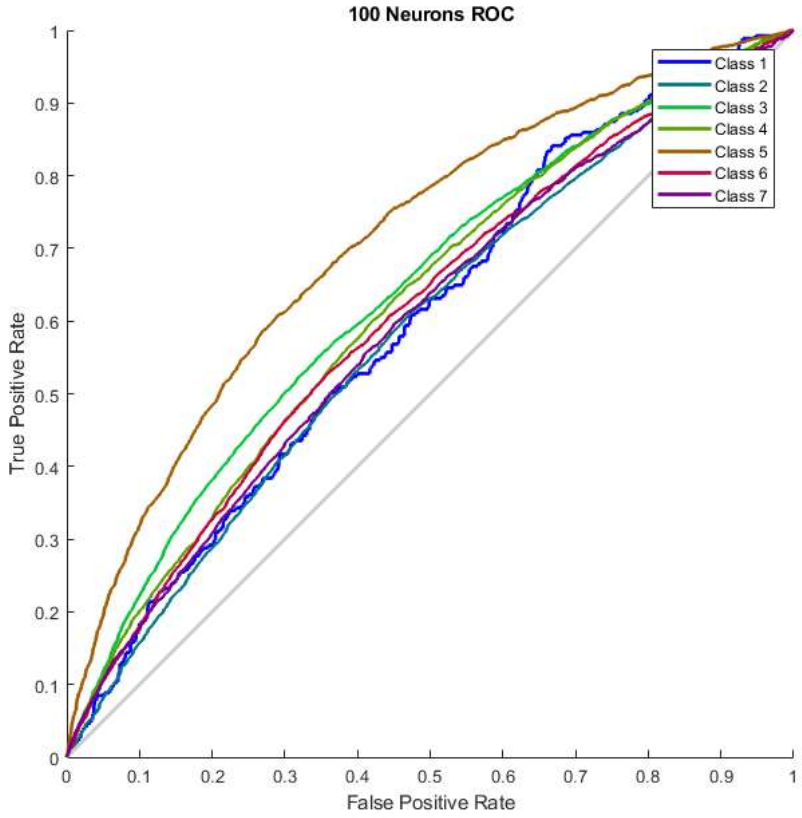

50 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

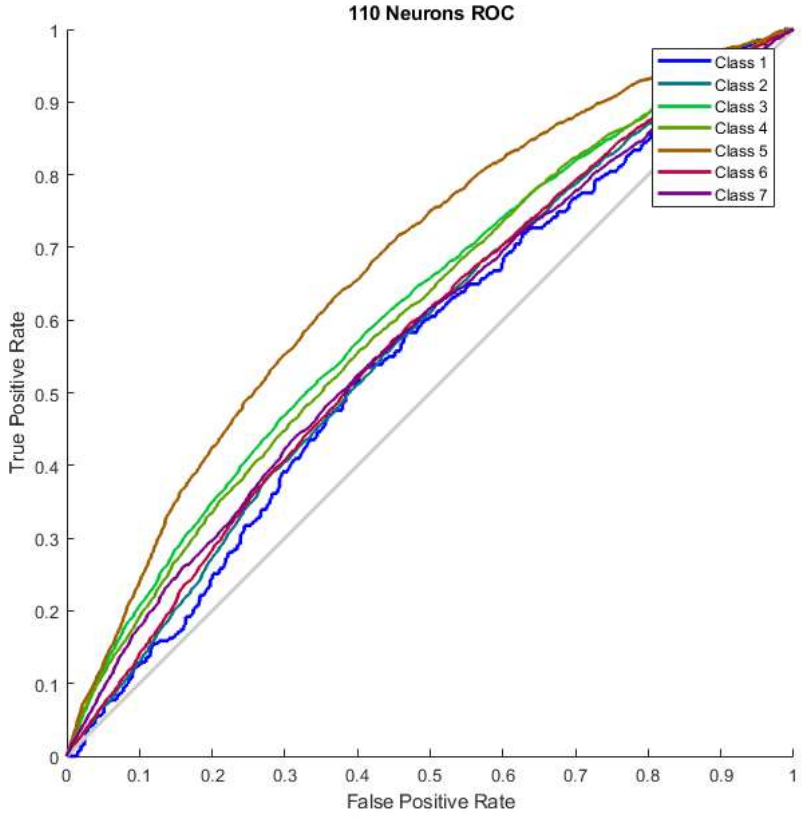

60 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

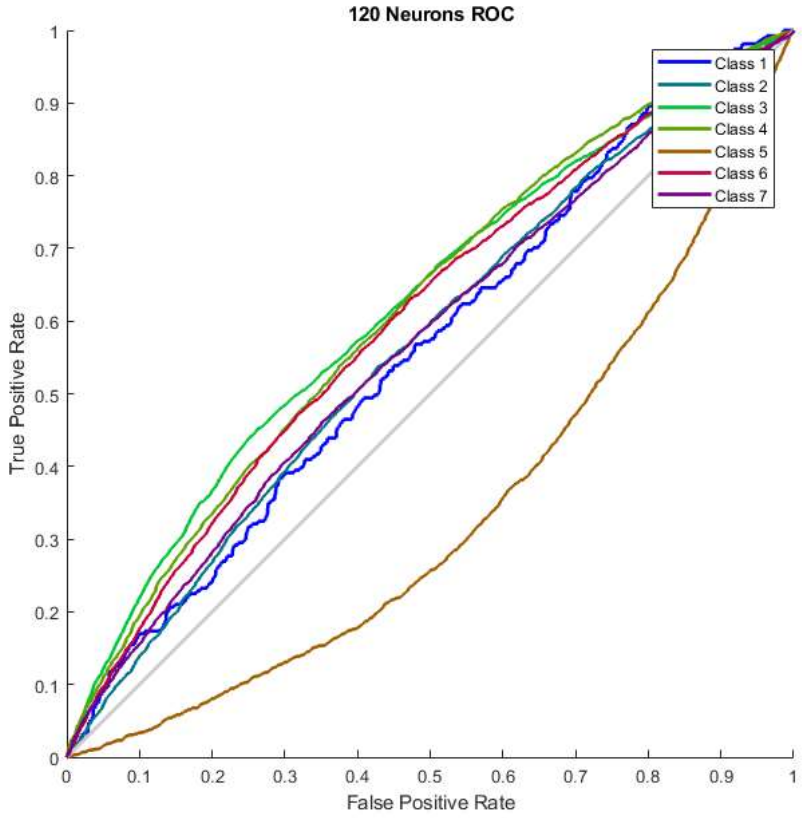

70 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

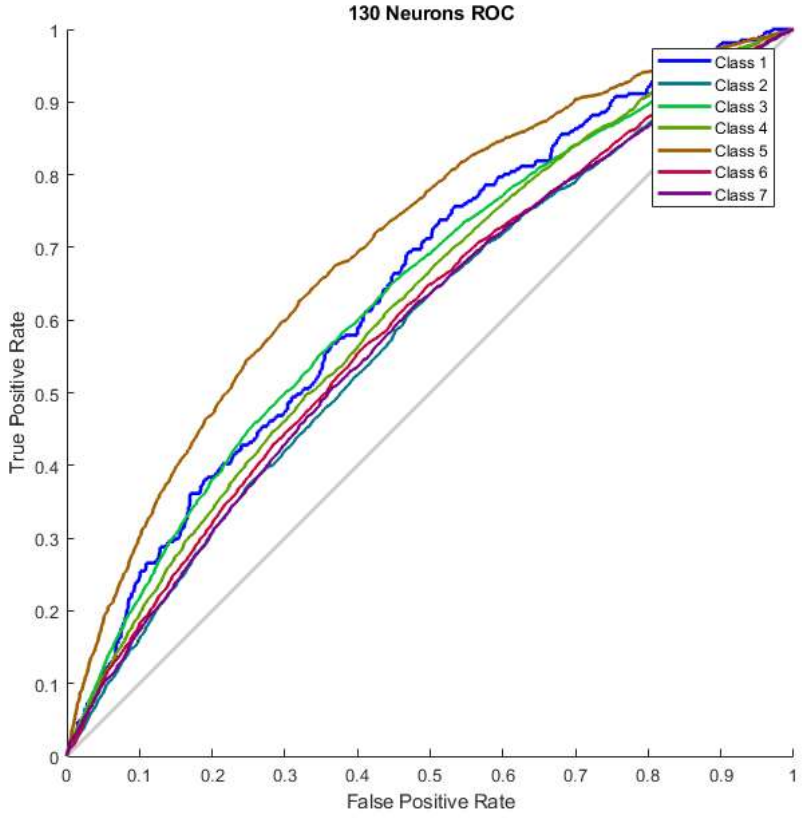

80 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

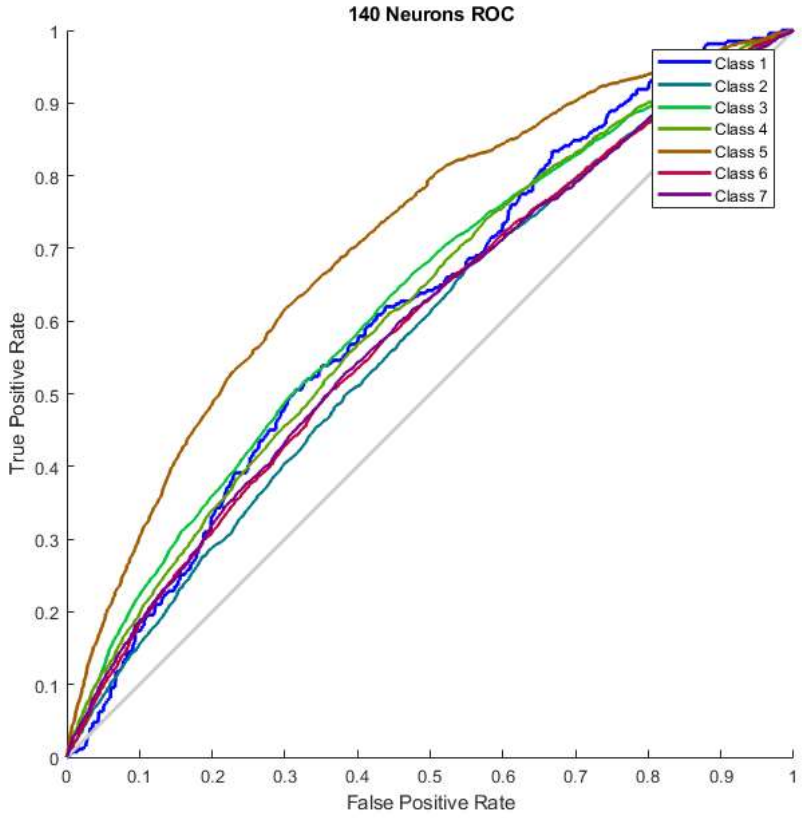

90 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

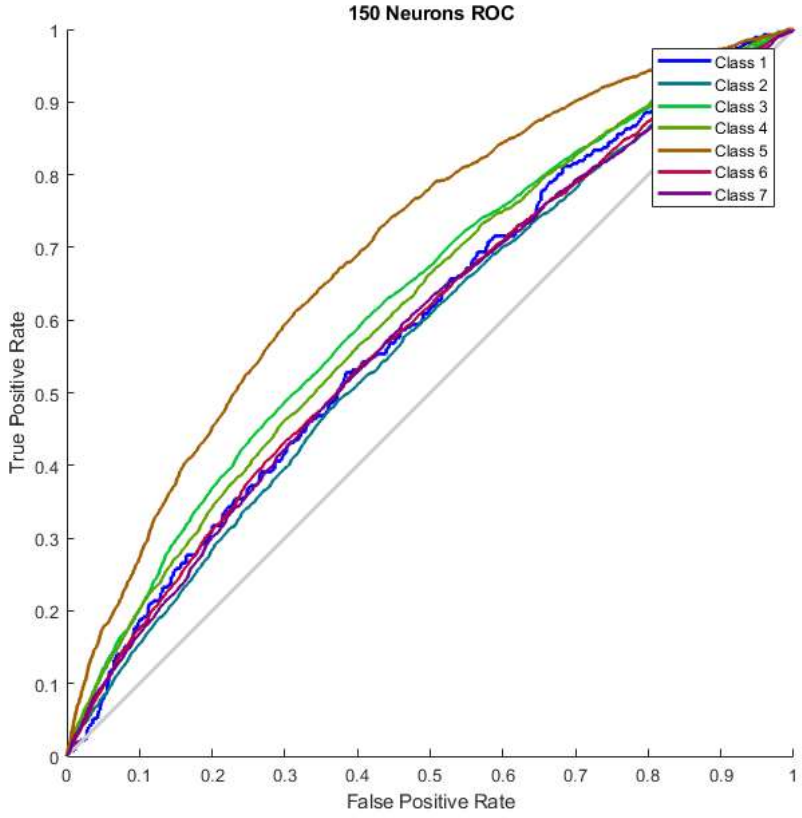

100 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

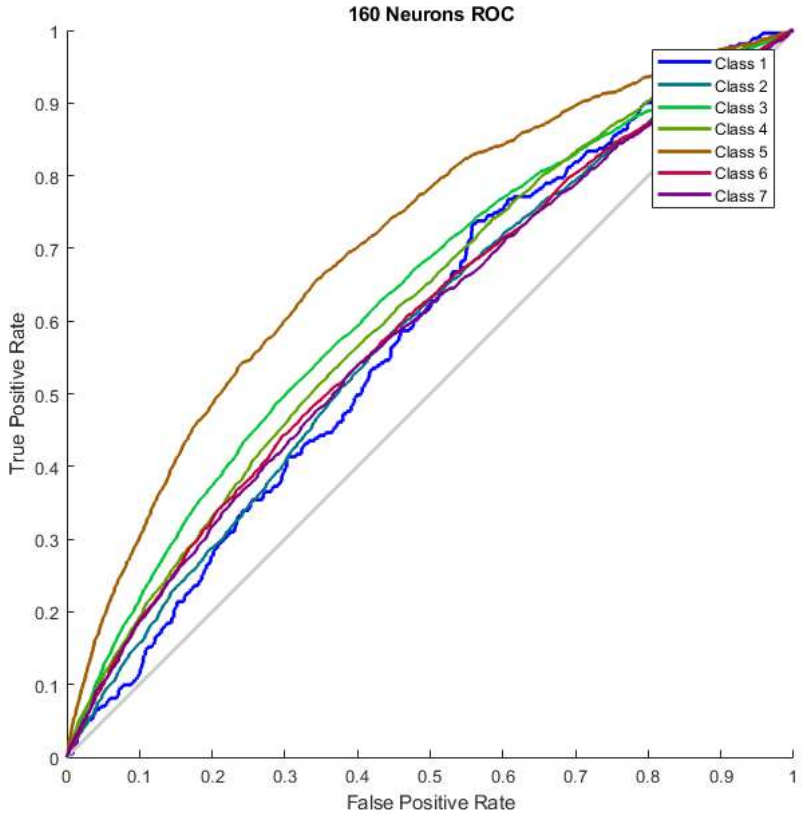

110 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

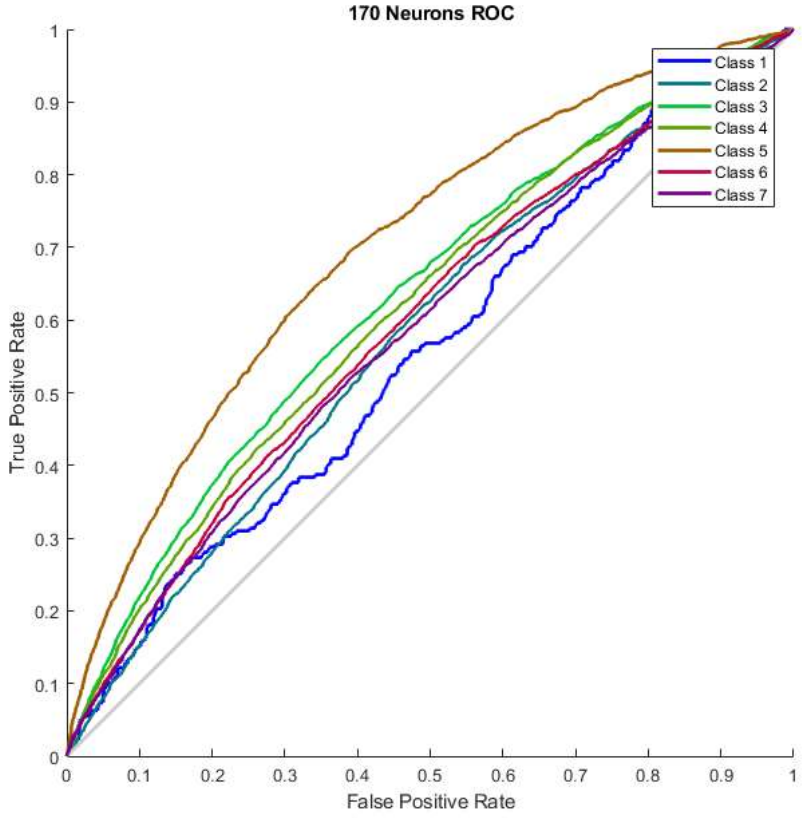

120 Neurons ROC
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
True Positive Rate
False Positive Rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

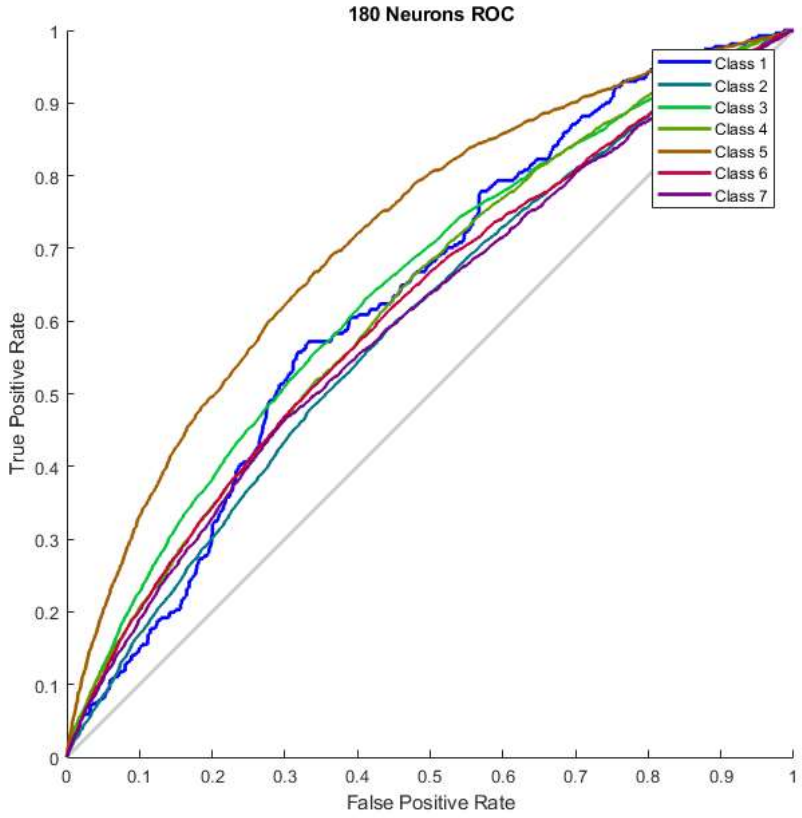

130 Neurons ROC
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
True Positive Rate
False Positive Rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

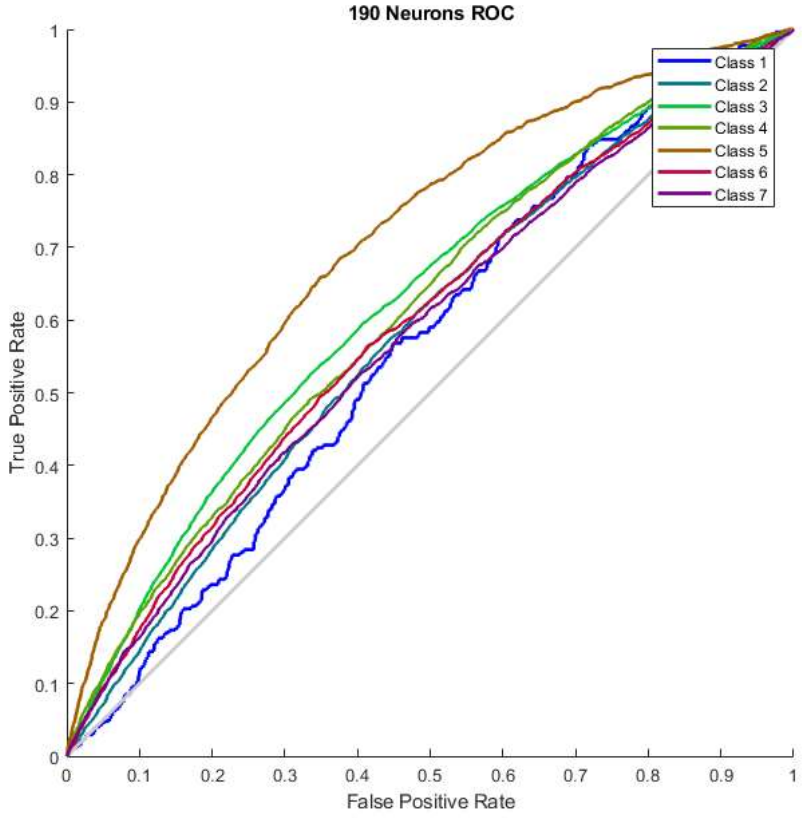

140 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

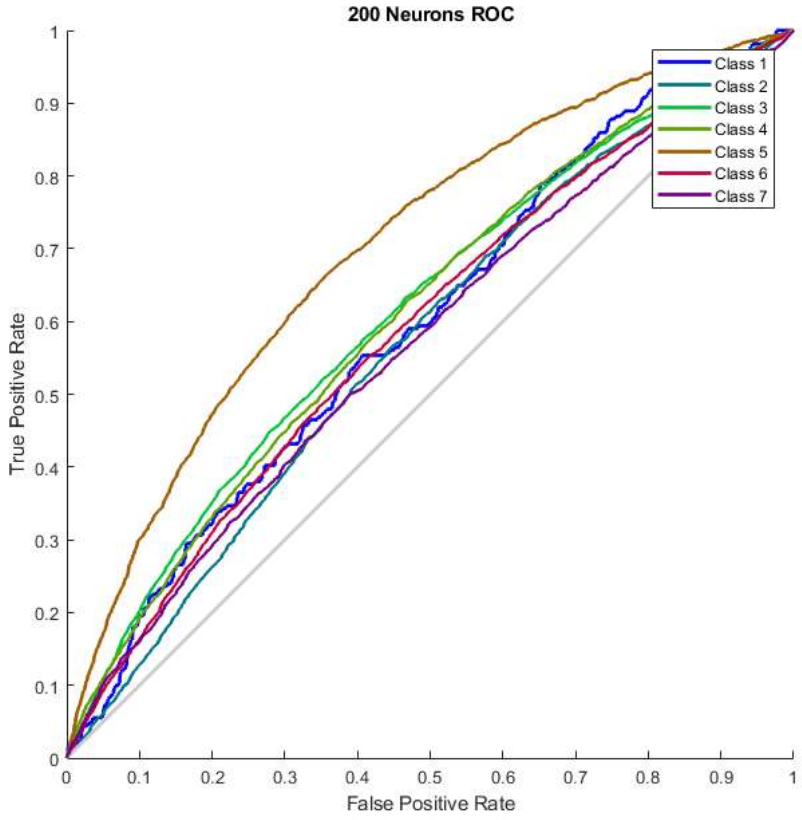

150 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


160 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


170 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


180 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


190 Neurons ROC
True Positive Rate
False Positive Rate
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



Published with MATLAB® R2017b

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
```

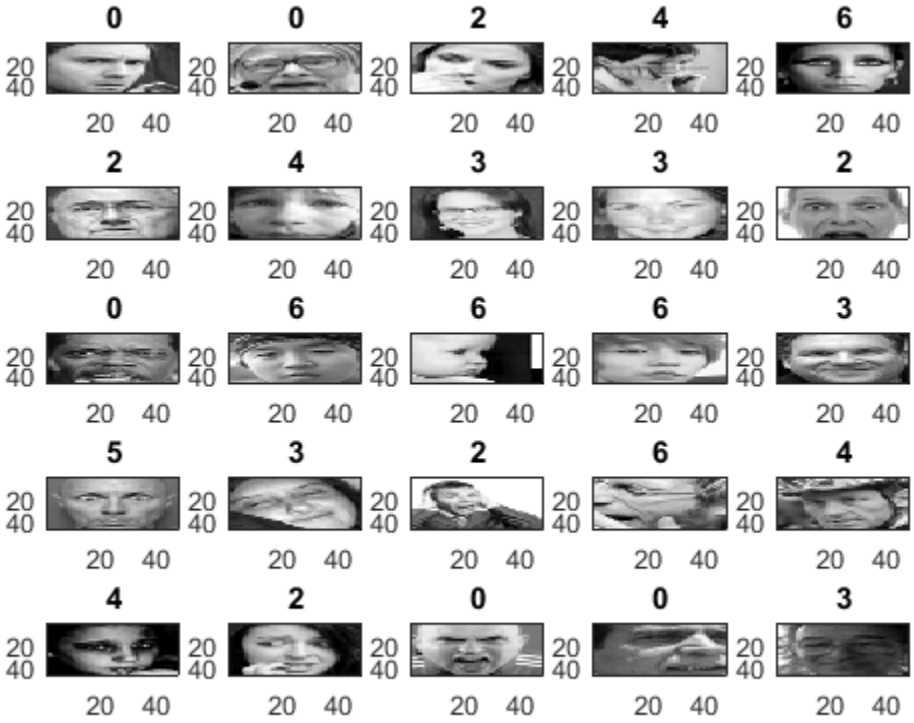
all data for testing

```
%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
```

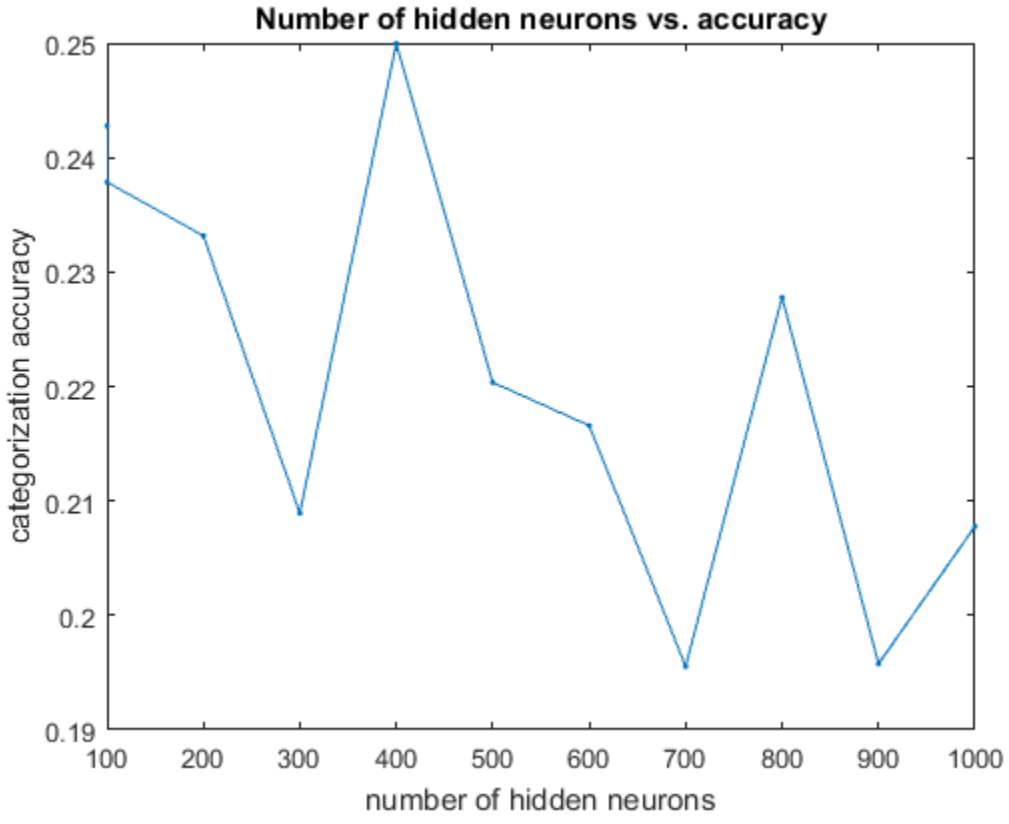
```
    0.0023    0.0410    0.0001    0.0007    0.0208
    0.0268    0.0001    0.0000    0.3337    0.9790
    0.0039    0.3958    0.0000    0.0001    0.0000
```

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

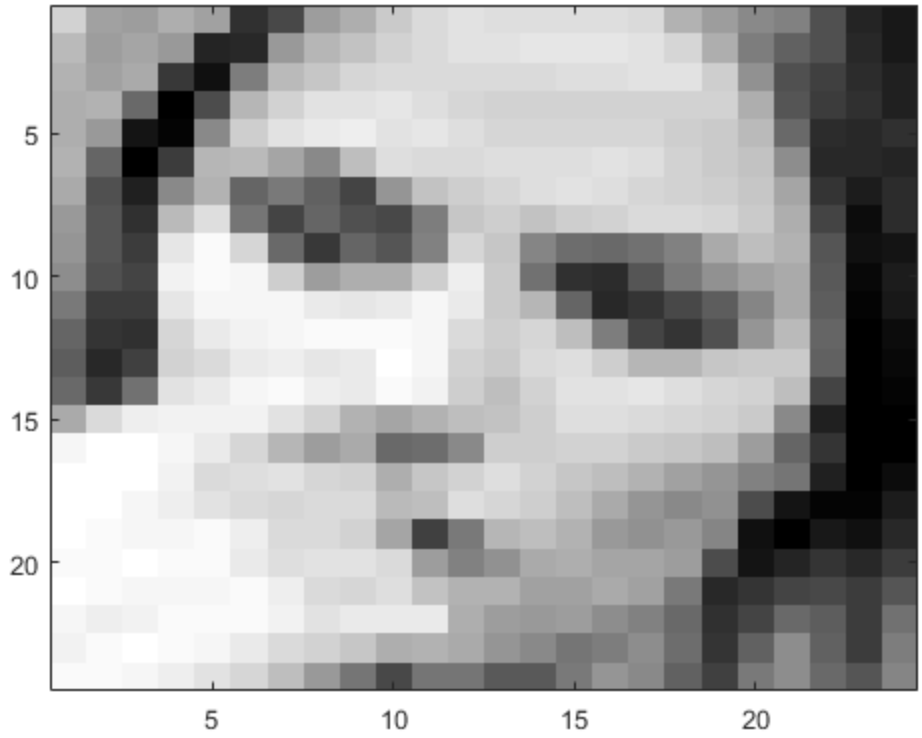
```
clc; clear all; close all;
```

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

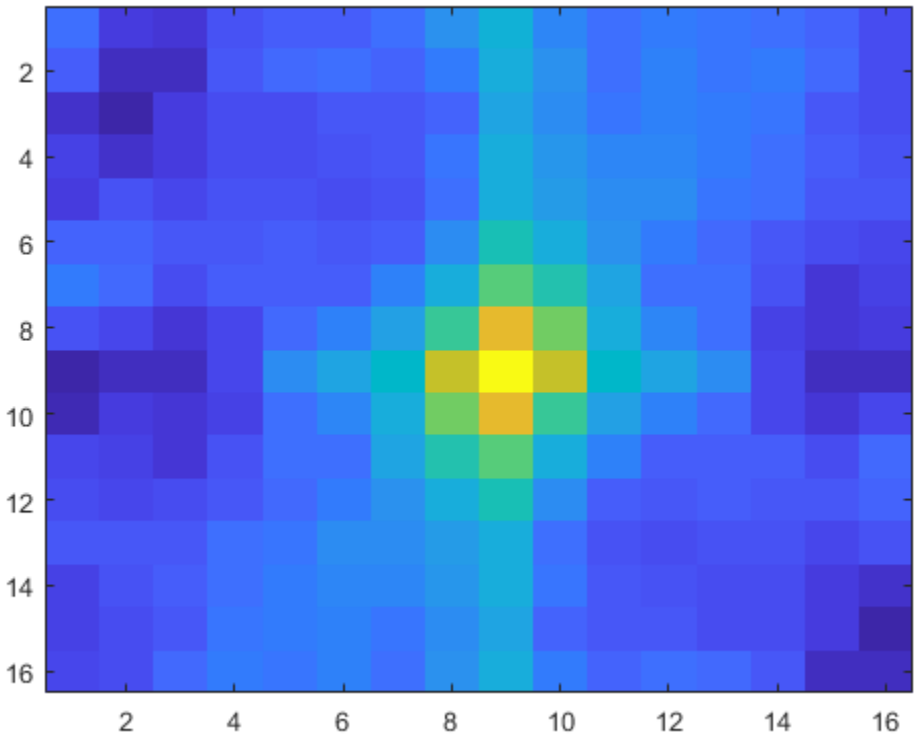
```
restoredP1 = log(abs(ifft2(im1_M*exp(1i*0)))+1);

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