

بسم الله الرحمن الرحيم

تمرین سوم شبکه عصبی ۱۳۹۴/۱۰/۱۵

خوب در ابتدا میخواهیم یک toolbox متنی برای شبکه های کانولوشن روی متلب نصب کنیم:

باید اول ببینیم که متلب کامپایر C رو میشناسه یا خیر، با دستور زیر در command window متلب این را می فهمیم:

mex -setup

برای من نشناخت به همین خاطر آخرین نسخه متلب رو یعنی 2015b رو نصب کردم تا 2015 visual studio رو بشناسه.

نصب **MatconveNet**:

## MatConvNet: CNNs for MATLAB

این برنامه در واقع به متلب اضافه میشه و میتونیم از شبکه های کانولوشنال در متلب بهره ببریم.

برای نصب اول آخرین ورژن رو از لینک زیر دانلود میکنیم:

<http://www.vlfeat.org/matconvnet/>

## MatconveNet معرفی نرم افزار

MatConvNet یکی از نرم افزارهای معروف جهت کار با Convolutional Neural Networks و Deep Learning است. وجه تمایز این نرم افزار در مقایسه با نرم افزارهای معروف دیگر در این حوزه، مانند Theano، Caffe و Torch راحتی نصب و استفاده و همچنین ایتنرفیس MATLAB است که آنرا برای استفاده در کارهای پژوهشی بسیار مناسب نموده است. سه نرم افزار دیگر، معمولاً در لینوکس نصب و پشتیبانی می شوند، اما MatConveNet به راحتی در ویندوز و دیگر سیستم عامل ها نصب می شود. با توجه به حجم محاسبات بالا برای آموزش شبکه های عمیق، هر چهار نرم افزار فوق GPU را پشتیبانی می کنند. ما در این فایل آموزشیبر روش نصب و کار کردن با نرم افزار MatConvNet متمرکز می شویم. آخرین نسخه نرم افزار و مستندات مربوط به آن از لینک زیر قابل مشاهده و Download می باشد.

<http://www.vlfeat.org/matconvnet/>

مراحل نصب:

پس از Download نرم افزار، مراحل زیر را جهت نصب نرم افزار در مطلب انجام دهید. ما نرم افزار را در محیط MATLAB2015b نصب نموده ایم. جهت نصب نرم افزار لازم است کامپایلر Visual Studio بر روی کامپیوتر شما نصب باشد و به نظر می رسد که ورژن Visual Studio باید قدیمی تر از MATLAB باشد. ما از Visual Studio2015 استفاده کرده ایم .

۱- ابتدا در خط فرمان مطلب دستور زیر را تایپ کنید :

mex -setup

در صورت نصب بودن Visual studio توضیحات مربوط به ظاهر می گردد.

۲- با استفاده از دستور `cd` در خط فرمان مطلب به پوشه MatConNet که دانلود کرده اید بروید و این پوشه را با دستور زیر به path مطلب اضافه نمایید.

```
addpath matlab
```

۳- در خط فرمان مطلب دستور `vl_compilenn` را وارد کنید. و صبر کنید تا مرحله کمپایل کامل شود.

۴- در خط فرمان مطلب دستور `vl_setupnn` را وارد کنید.

۵- جهت تست نرم افزار و اطمینان از صحت نصب، دستور `vl_testnn` را در خط فرمان مطلب وارد نمایید.

خلاصه مراحل فوق به صورت زیر است:

1. `mex-setup`
2. `addpath matlab`
3. `vl_compilenn`
4. `vl_setupnn`
5. `vl_testnn`

کار با نرم افزار

در ادامه طی سه مثال کاربردی با برخی از قابلیت های نرم افزار آشنا می شویم.

مثال (۱) تست شبکه آموزش داده شده GoogleNet :

در این مثال با یک برنامه ساده، با قابلیت ها و قدرت شبکه GoogleNet آشنا می شویم. برای اینکار لازم است در ابتدا، شبکه فوق را دانلود کنیم. لینک دانلود شبکه به صورت زیر است:

<http://www.vlfeat.org/matconvnet/models/imagenet-vgg-f.mat>

حجم این فایل حدود ۲۴۰ مگابایت است و ساختار شبکه و وزن ها را در خود ذخیره کرده است بهتر است بعد از دانلود فایل، آنرا در پوشه که در مرحله ۲ نصب مسیر آنرا به مطلب اضافه کرده اید کپی نمایید تا برای استفاده های بعدی در دسترس باشد. شبکه GoogleNet یک شبکه ۲۲ لایه است که با ۲/۱ میلیون عکس رنگی ۲۲۴ در ۲۲۴ آموزش داده شده است. خروجی شبکه شامل ۱۰۰۰ دسته می باشد. جهت آشنایی بیشتر با این شبکه می توانید مقاله زیر را مطالعه کنید.

going deeper with convolutions [CVPR2015]

پس از دانلود شبکه با استفاده از برنامه زیر شبکه را لود و تست نمایید. (قبلش باید MatConNet رو نصب (کمپایل) کرده باشیم)

```
clc
clear
close all;
% add MatConvNet installed directory
```

با ۲ خط کد زیر دیگر لازم نیست در هر بار اجرا دایرکتوری که MatConvNet در آن نصب شده است به صورت دستی وارد کنیم: (این ۲ خط کد این کار را قبل از اجرای سایر قسمت ها انجام میدهد)

```
cd F:\Documents\MATLAB\MatConvNet\matconvnet-1.0-beta17;
```

```

addpath matlab;
%add pre MatConvNet setup
vl_setupnn;
% load the pre-trained CNN
net = load('imagenet-vgg-f.mat');

% load and preprocess an image
im = imread('13.jpg') ;
im_ = single(im) ; % note: 0-255 range
im_ = imresize(im_, net.meta.normalization.imageSize(1:2))
;
im_ = im_ - net.meta.normalization.averageImage ;
% run the CNN
res = vl_simplenn(net, im_) ;

% show the classification result
scores = squeeze(gather(res(end).x)) ;
[bestScore, best] = max(scores);
figure(1) ; clf ; imagesc(im) ;
title(sprintf('%s (%d), score %.3f',...
net.meta.classes.description{best}, best, bestScore)) ;

```

مهمترین دستور از برنامه فوق، دستور `vl_simplenn` است که شبکه و یک نمونه عکس می گیرد و خروجی را تولید می کند. این دستور معادل دستور `sim` از `toolbox` شبکه عصبی مطلب است.

تمرین: ۱۰ عکس از محیط اطراف خود بگیرید و به جای عکس `peoors.png` به شبکه بدهید. مشاهدات خود را گزارش دهید.

## مثال ۲ – Transfer Learning

پس از آشنا شدن با قدرت شبکه `googleNet`، قصد داریم از این شبکه برای کلاسه بندی تصاویر بر اساس نیاز خود استفاده کنیم. استفاده از شبکه های از پیش آموزش داده شده بر روی داده های دیگر (غیر از داده های آموزشی) تحت عنوان `Transfer Learning` مطرح می شود. مثلاً ممکن است شخصی، شبکه ای برای تشخیص اسب از زرافه آموزش داده باشد و شما قصد داشته باشید از این شبکه برای تشخیص سگ از گربه استفاده کنید.

مزیت این روش این است که معمولاً با داده های آموزشی به مراتب کمتر از داده های آموزشی شبکه آموزش داده شده اولیه، می توانیم شبکه قدرتمندی جهت مساله خود آموزش دهیم. روش کار به این صورت است که از شبکه آموزش داده شده به عنوان `Feature Extractor` استفاده می کنیم. یعنی داده های خود را به شبکه وارد می کنیم و خروجی لایه یکی مانده به آخر را به عنوان بردار ویژگی استفاده می کنیم. با در اختیار داشتن ویژگی های مناسب، یک شبکه ساده برای مساله خود طراحی می کنیم و آن را آموزش می دهیم. در ادامه، به عنوان مثال شبکه

GoogleNet را به عنوان Feature Extractor جهت تشخیص هواپیمای مسافری از جنگنده بکار خواهیم گرفت. ما حدود ۷۰ عکس از هریک از کلاس های هواپیمای جنگنده و مسافری را جمع آوری کردیم. هریک از تصاویر را به شبکه وارد کردیم و خروجی لایه ماقبل آخر را به عنوان بردار ویژگی برای هر عکس ذخیره کردیم سپس یک SVM خطی(بدون کرنل) برای کلاسه بندی آموزش دادیم.

```
clc
clear
close all;
% load the pre-trained CNN
cnnModel.net = load('imagenet-vgg-f.mat');
%% Load images from folder
% Use imageSet to load images stored in pet_images folder
imset = imageSet('Aircraft_images','recursive');
% Preallocate arrays with fixed size for prediction
imageSize = cnnModel.net.normalization.imageSize;
trainingImages = zeros([imageSize
sum([imset(:).Count]),'single']);
% Load and resize images for prediction
counter = 0;
for ii = 1:numel(imset)
for jj = 1:imset(ii).Count
counter = counter +1;
trainingImages(:, :, :, counter) =
imresize(single(read(imset(ii),jj)),imageSize(1:2));
end
end
% Get the image labels
trainingLabels = getImageLabels(imset);
summary(trainingLabels) % Display class label distribution
cnnModel.info.opts.batchSize = 30;
cnnFeatures = cnnPredict(cnnModel,trainingImages);
%% Train a classifier using extracted features
% Here I train a linear support vector machine (SVM)
classifier.
svmdl = fitcsvm(cnnFeatures,trainingLabels);
% Perform crossvalidation and check accuracy
cvmdl = crossval(svmdl,'KFold',10);
fprintf('kFold CV accuracy: %2.2f\n',1-cvmdl.kfoldLoss)
```

فایل های مربوطه ضمیمه این گزارش هستند. با این روش ما با دقت ۹۳ درصد رسیدیم.

به دلخواه دو کلاس انتخاب کنید، تصاویری از کلاسها جمع آوری کنید و پروسه فوق را برای داده های خودتان تکرار کنید. نتایج خود را گزارش دهید.

تمرین ۲ به جای SVM از یک نرون ساده همرا با رگولاریزه کر دن استفاده کنید نتایج خود را با مرحله قبل مقایسه کنید.

مثال ۳ - روش ساخت یک شبکه با استفاده از نرم افزار MatConvNet

در مراحل قبل روش استفاده از شبکه های از پیش آموزش داده شده را بررسی کردیم. در این مرحله روش طراحی یک شبکه از پایه و آموزش آن را بررسی می کنیم. شبکه خود را بر روی داده های MNIST تست می کنیم. مراحل انجام کار در برنامه زیر آمده است:

```
clear;
close all;
clc
% -----
%
options
% -----
opts.dataDir = fullfile('data','mnist') ;
opts.train.batchSize = 100;
opts.train.numEpochs = 3 ;
opts.train.learningRate = 0.001 ;
% -----
%
Prepare data
% -----
imdb = getMnistImdb(opts);
f=1/100 ;
net.layers = {} ;
net.layers{end+1} = struct('type', 'conv', ...
    'weights', {{f*randn(5,5,1,20, 'single')}, zeros(1, 20,
    'single')}}}, ...
    'stride', 1, ...
    'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
    'method', 'max', ...
    'pool', [2 2], ...
    'stride', 2, ...
```

```

'pad', 0) ;
net.layers{end+1} = struct('type', 'conv', ...
'weights', {{f*randn(5,5,20,50,
'single'),zeros(1,50,'single')}}}, ...
'stride', 1, ...
'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
'method', 'max', ...
'pool', [2 2], ...
'stride', 2, ...
'pad', 0) ;
net.layers{end+1} = struct('type', 'conv', ...
'weights', {{f*randn(4,4,50,500, 'single'),
zeros(1,500,'single')}}}, ...
'stride', 1, ...
'pad', 0) ;
net.layers{end+1} = struct('type', 'relu') ;
net.layers{end+1} = struct('type', 'conv', ...
'weights', {{f*randn(1,1,500,10, 'single'),
zeros(1,10,'single')}}}, ...
'stride', 1, ...
'pad', 0) ;
net.layers{end+1} = struct('type', 'softmaxloss') ;
% -----
%
%
Train
% -----
[net, info] = cnn_train(net, imdb, @getBatch, ...
    opts.train, ...
'val', find(imdb.images.set == 3)) ;

```

تمرین: برنامه فوق را اجرا کنید و نتایج بدست آمده را با نتایج تکلیف پیش مقایسه کنید.

## کار با تصاویر در متلب

# **Working with Images in MATLAB**

## **Teacher's Day Workshop**

**School of Computing and Communications**

**December 2013**

## 1. Work with Images in MATLAB

Digital image is composed of a two or three dimensional matrix of pixels. Individual pixels contain a number or numbers representing what grayscale or color value is assigned to it. Color pictures generally contain three times as much data as grayscale pictures, depending on what color representation scheme is used. Therefore, color pictures take three times as much computational power to process.

MATLAB can import/export several image formats:

- BMP (Microsoft Windows Bitmap)
- GIF (Graphics Interchange Files)
- HDF (Hierarchical Data Format)
- JPEG (Joint Photographic Experts Group)
- PCX (Paintbrush)
- PNG (Portable Network Graphics)
- TIFF (Tagged Image File Format)
- XWD (X Window Dump)
- raw-data and other types of image data.



## 2.1 Read and Display an Image

You can read standard image files by using the *imread* function. The type of data returned by *imread* depends on the type of image you are reading. For example, read *image1.jpg* by typing (the image can be downloaded using the following link. <http://crin.eng.uts.edu.au/~rob/image1.jpg>, and then can be copied into the current folder):

```
A = imread('image1.jpg');
```

which will store *image1.jpg* in a matrix named A.

Now display the image using the *imshow* function. For example, type:

```
imshow(A);
```

## 2.2 Grayscale Images

A grayscale image is a data matrix whose values represent intensities within some range. MATLAB stores a grayscale image as an individual matrix, with each element of the matrix corresponding to one image pixel.

```
B = rgb2gray(A);
```

Now display the image by typing:

```
imshow(B);
```

## 2.3 Write the Image to a Disk File

To write the newly adjusted image B to a disk file, use the *imwrite* function. If you include the filename extension '.png', the *imwrite* function writes the image to a file in Portable Network Graphics (PNG) format, but you can specify other formats. For example, type:

```
imwrite (B, 'image2.png');
```

## 2.4 Check the Contents of the Newly Written File

To see what *imwrite* wrote to the disk file, use the *imfinfo* function.

```
imfinfo('image2.png')
```

The *imfinfo* function returns information about the image in the file, such as its format, size, width, and height.

```

ans =
Filename: 'image2.png' FileModDate: '12-Nov-2013 10:43:31'
FileSize: 52936 Format: 'png'
FormatVersion: []
Width: 350
Height: 350
BitDepth: 8 ColorType: 'grayscale'
FormatSignature: [137 80 78 71 13 10 26 10]
Colormap: []
Histogram: []
InterlaceType: 'none'
Transparency: 'none'
SimpleTransparencyData: []
BackgroundColor: []
RenderingIntent: []
Chromaticities: []
Gamma: []
XResolution: []
YResolution: []
ResolutionUnit: []
XOffset: []
YOffset: []
OffsetUnit: []
SignificantBits: []
ImageModTime: '11 Nov 2013 23:43:31 +0000'
Title: []
Author: []
Description: []
Copyright: []
CreationTime: []
Software: []
Disclaimer: []
Warning: []
Source: []
Comment: []
OtherText: []

```

## 2.5 Resize an Image

To resize an image, use the *imresize* function. When you resize an image, you specify the image to be resized and the magnification factor. To enlarge an image, specify a magnification factor greater than 1. To reduce an image, specify a magnification factor between 0 and 1.

```

imshow(B);

C = imresize(B,1.5);

```

```
figure
imshow(C);
```

```
C = imresize(B,0.5);
figure
imshow(C);
```

You can specify the size of the output image by passing a vector that contains the number of rows and columns in the output image. If the specified size does not produce the same aspect ratio as the input image, the output image will be distorted.

```
C = imresize(B,[300,150]);
figure
imshow(C);
```

This example creates an output image with 300 rows and 150 columns.

## 2.6 Rotate an Image

To rotate an image, use the *imrotate* function. When you rotate an image, you specify the image to be rotated and the rotation angle, in degrees. If you specify a positive rotation angle, *imrotate* rotates the image counterclockwise; if you specify a negative rotation angle, *imrotate* rotates the image clockwise.

```
C = imrotate(B,35);
figure
imshow(C);
```

```
C = imrotate(B,-20);
figure
imshow(C);
```

## 2.7 Crop an Image

Cropping an image means creating a new image from a part of an original image. To crop an image using the Image Viewer, use the **Crop Image** tool or use the *imcrop* function.



### Using the Crop Image Tool:

By default, if you close the Image Viewer, it does not save the modified image data. To save the cropped image, you can use the Save As option from the Image Viewer File menu to store

the modified data in a file or use the Export to Workspace option to save the modified data in the workspace variable. To use the Crop Image tool, follow this procedure:

- 1) View an image in the Image Viewer.


```
imtool(A);
```

- 2) Start the Crop Image tool by clicking **Crop Image**  in the Image Viewer toolbar or selecting **Crop Image** from the Image Viewer Tools menu. (Another option is to open a figure window with `imshow` and call `imcrop` from the command line.) When you move the pointer over the image, the pointer changes to cross hairs .
- 3) Define the rectangular crop region, by clicking and dragging the mouse over the image. You can fine-tune the crop rectangle by moving and resizing the crop rectangle using the mouse.
- 4) When you are finished defining the crop region, perform the crop operation. Double-click the left mouse button or right-click inside the region and select **Crop Image** from the context menu. The Image Viewer displays the cropped image.
- 5) To save the cropped image, use the Save as option or the Export to Workspace option on the Image Viewer File menu.

Now display the image using the `imshow` function.

### **Using the `imcrop` Function:**

By using the `imcrop` function, you can specify the crop region interactively using the mouse or programmatically by specifying the size and position of the crop region.

This example illustrates an interactive syntax. The example reads an image into the MATLAB workspace and calls `imcrop` specifying the image as an argument. `imcrop` displays the image in a figure window and waits for you to draw the crop rectangle on the image. When you move the pointer over the image, the shape of the pointer changes to cross hairs . Click and drag the pointer to specify the size and position of the crop rectangle. You can move and adjust the size of the crop rectangle using the mouse. When you are satisfied with the crop rectangle, double-click to perform the crop operation, or right-click inside the crop rectangle and select Crop Image from the context menu. `imcrop` returns the cropped image.

```
C = imcrop(A);  
  
figure  
  
imshow(C);
```

**Raw MATLAB:** For advanced users, the native MATLAB commands can be used. You can specify the size and position of the crop rectangle as parameters when you call *imcrop*. Specify the crop rectangle as a four-element position vector, [xmin ymin width height].

In this example, you call *imcrop* specifying the image to crop, A, and the crop rectangle. *imcrop* returns the cropped image in D.

```
D = imcrop(A,[160 140 110 180]);  
figure  
imshow(D);
```

## 2.8 Getting Image Pixel Values

You can get information about specific image pixels such as RGB values. Type:

```
A(2,15,:)
```

which returns the RGB (red, green, and blue) color values of the pixel (2,15). R=66; G= 88; B= 174.

```
ans(:,:,1) =  
    66  
  
ans(:,:,2) =  
    88  
  
ans(:,:,3) =  
   174
```

Now try:

```
A(40:100,10:20,:)
```

You can also use the *impixel* function which will determine the values of one or more pixels in an image and return the values in a variable. Select the pixels interactively using a mouse. *impixel* returns the value of specified pixels in a variable in the MATLAB workspace.

The following example illustrates how to use *impixel* to get pixel values.

### 1) Display an image.

```
imshow(A);
```

### 2) Call *impixel*. When called with no input arguments, *impixel* associates itself with the image in the current axes.

```
vals = impixel
```

3) Select the points you want to examine in the image by clicking the mouse. *impixel* places a star at each point you select.



4) When you are finished selecting points, press Return. *impixel* returns the pixel values in an n-by-3 array, where n is the number of points you selected. The stars used to indicate selected points disappear from the image.

```
vals =  
  
    46     71    155  
    80     96    184  
    95    107    193
```

## 2.9 Changing Image Pixel Values

You can change the values of specific image pixels. Type:

```
A(40:100,10:20,:) = 0;
```

```
figure
```

```
imshow(A);
```

which changes the colors of the selected pixels into black color.

Now try:

```
A(40:100,10:20,:) = 255;
```

```
figure
imshow(A);
```

## 2.10 Image Intensity Adjustment

Image intensity adjustment is used to improve an image, Read *image1.jpg* again.

```
A = imread('image1.jpg');
```

Multiply the image pixels values by two.

```
E = A.*2;
figure
imshow(E);
```

Now try:

```
F = A.*0.75;
figure
imshow(F);
```

Then, try:

```
F = A.*7.5;
figure
imshow(F);
```

## 2.11 Detecting Edges Using the edge Function

In an image, an edge is a curve that follows a path of rapid change in image intensity. Edges are often associated with the boundaries of objects in a scene. Edge detection is used to identify the edges in an image. To find edges, you can use the `edge` function. This function looks for places in the image where the intensity changes rapidly, using one of these two criteria:

- Places where the first derivative of the intensity is larger in magnitude than some threshold.
- Places where the second derivative of the intensity has a zero crossing.

`edge` provides a number of derivative estimators, each of which implements one of the definitions above. For some of these estimators, you can specify whether the operation should be sensitive to horizontal edges, vertical edges, or both. `edge` returns a binary image containing 1's where edges are found and 0's elsewhere.

The most powerful edge-detection method that edge provides is the `Canny` method. The Canny method differs from the other edge-detection methods in that it uses two different thresholds (to detect strong and weak edges), and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.

The following example illustrates the power of the `Canny` edge detector by showing the results of applying the `Sobel` and `Canny` edge detectors to the same image:

1) Read the image and display it. `G`

```
= imread('image2.png');  
imshow(G);
```

2) Apply the `Sobel` and `Canny` edge detectors to the image and display them.

```
BW1 = edge(G, 'sobel');  
BW2 = edge(G, 'canny');
```

```
figure  
imshow(BW1);
```

```
figure  
imshow(BW2);
```

## 2.12 Removing Noise from an Image

Digital images are prone to a variety of types of noise. Noise is the result of errors in the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. There are several ways that noise can be introduced into an image, depending on how the image is created. For example:

- If the image is scanned from a photograph made on film, the film grain is a source of noise. Noise can also be the result of damage to the film, or be introduced by the scanner itself.
- If the image is acquired directly in a digital format, the mechanism for gathering the data (such as a CCD detector) can introduce noise.
- Electronic transmission of image data can introduce noise.

You can use linear filtering to remove certain types of noise. Certain filters, such as averaging or Gaussian filters, are appropriate for this purpose. For example, an averaging filter is useful for removing grain noise from a photograph. Because each pixel gets set to the average of the pixels in its neighborhood, local variations caused by grain are reduced.



Median filtering is similar to using an averaging filter, in that each output pixel is set to an average of the pixel values in the neighborhood of the corresponding input pixel. However, with median filtering, the value of an output pixel is determined by the median of the neighborhood pixels, rather than the mean. The median is much less sensitive than the mean to extreme values (called outliers). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image.

The following example compares the use of a linear Gaussian filter and a median filter to remove salt and pepper noise for the same image:

**1) Read the image and display it.**

```
H = imread('image2.png');  
imshow(H);
```

**2) Add salt and pepper noise to the image and then display it.**

```
I = imnoise(H, 'salt & pepper', 0.02);  
  
figure  
imshow(I);
```

**3) Filter the noisy image using a linear Gaussian filter.**

- Create a Gaussian filter using the `fspecial` function.

```
filter = fspecial('gaussian', [3 3], 0.5);
```

- Filter the image using the created filter and then display the filtered image.

```
J = imfilter(I, filter, 'replicate');  
  
figure  
imshow(J);
```

**4) Filter the noisy image using a median filter by applying the `medfilt2` function and then display the filtered image.**

```
K = medfilt2(I, [3 3]);  
  
figure  
imshow(K);
```

## 2. Getting Help in MATLAB

For reference information about any of the functions, type in the MATLAB command window:

```
help functionname
```

For example:

```
help imread
```

یک سری کاربرد دیگر:



A tutorial on  
intensity transforms

## ۱- سوال یک

برای پاسخ به این سوال از دو شبکه آموزش دیده گوگل نت را استفاده کردم، که دومی شبکه با عمق زیاد (۴۳ لایه) می باشد و نتایج هر دو شبکه را در ادامه مشاهده میکنید:

نتیجه شبکه با ۴۳ لایه:

imagenet-vgg-verydeep-19.mat: monarch, monarch butterfly, milkweed butterfly, Danaus plexippus (324), score 0.798



همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۰,۷۸۹)

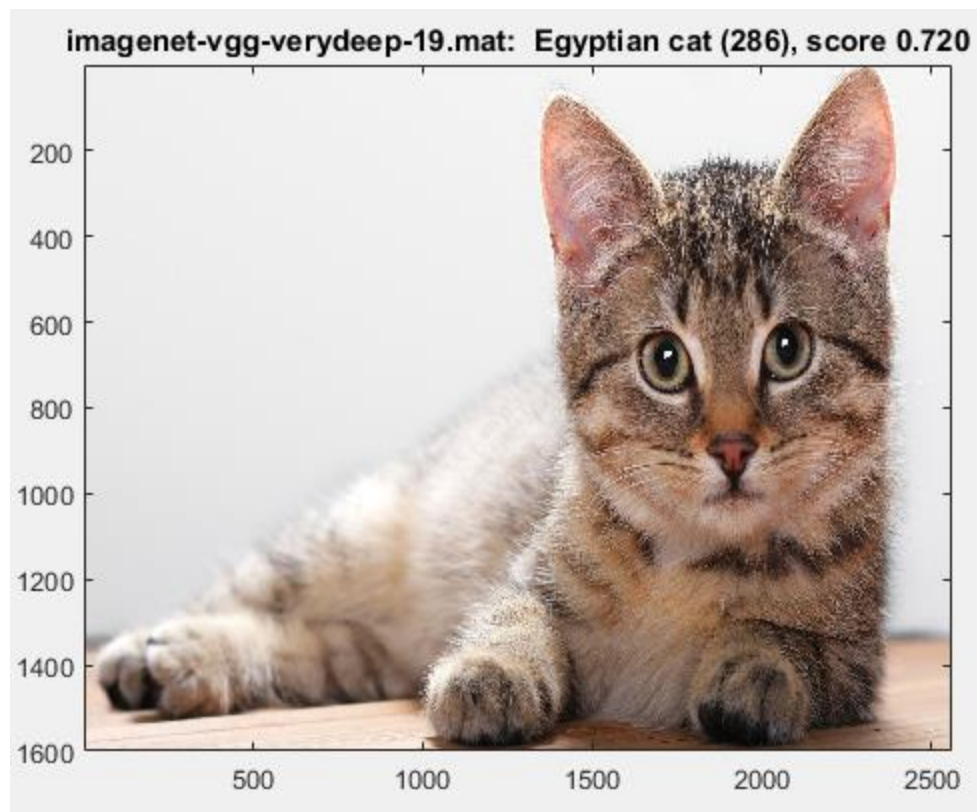
نتیجه شبکه معمولی: (۲۱ لایه)

imagenet-vgg-f.mat: monarch, monarch butterfly, milkweed butterfly, Danaus plexippus (324), score 0.368



این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد (امتیاز ۰,۳۶۸)

نتیجه شبکه با ۱۹ لایه:



همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۰,۷۲۰)

نتیجه شبکه معمولی:



این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد(امتیاز ۰,۴۱۲)

نتیجه شبکه با ۱۹ لایه:



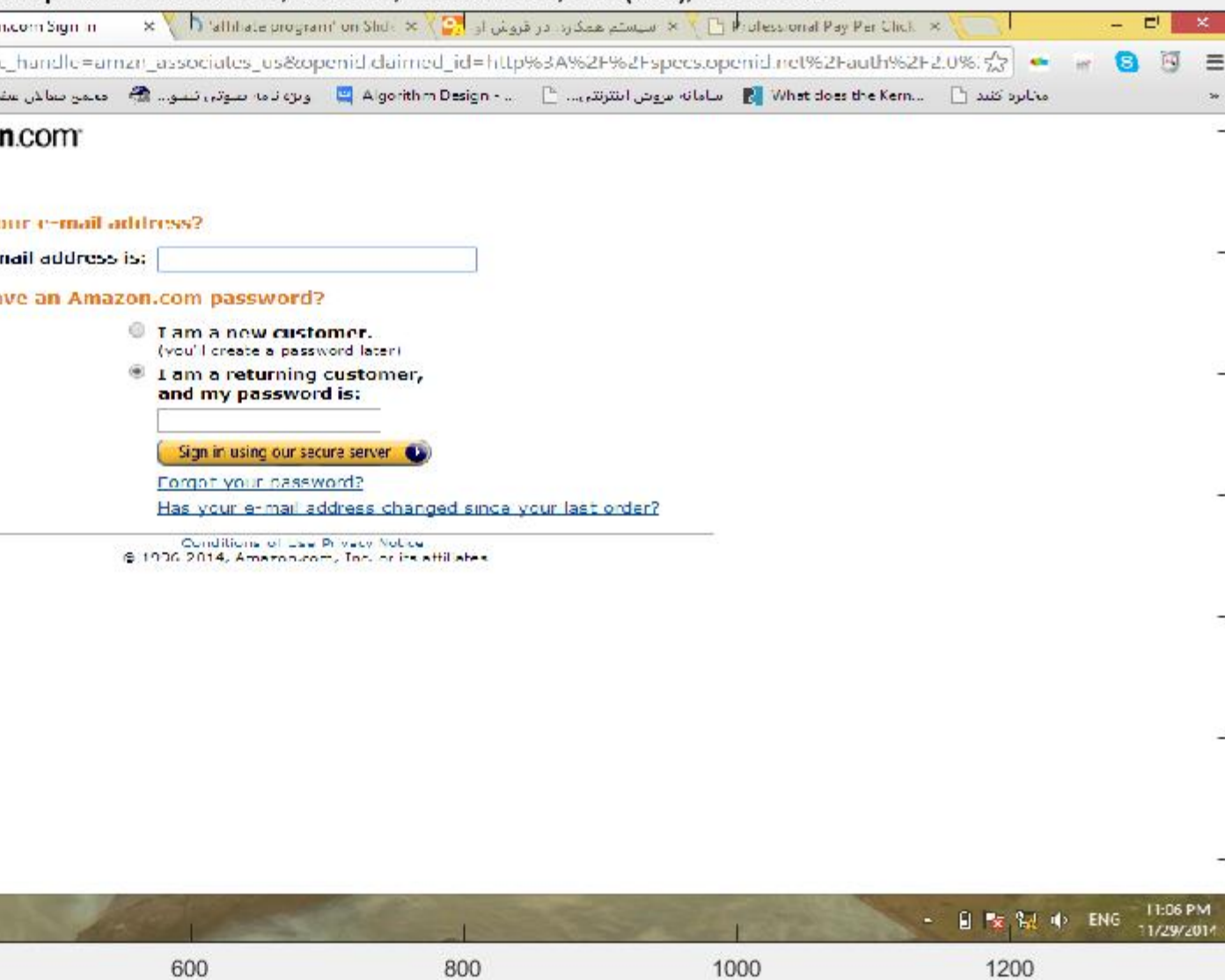
نتیجه شبکه معمولی:

این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد (امتیاز: ۰,۳۶۸).

نتیجه شبکه با ۱۹ لایه:



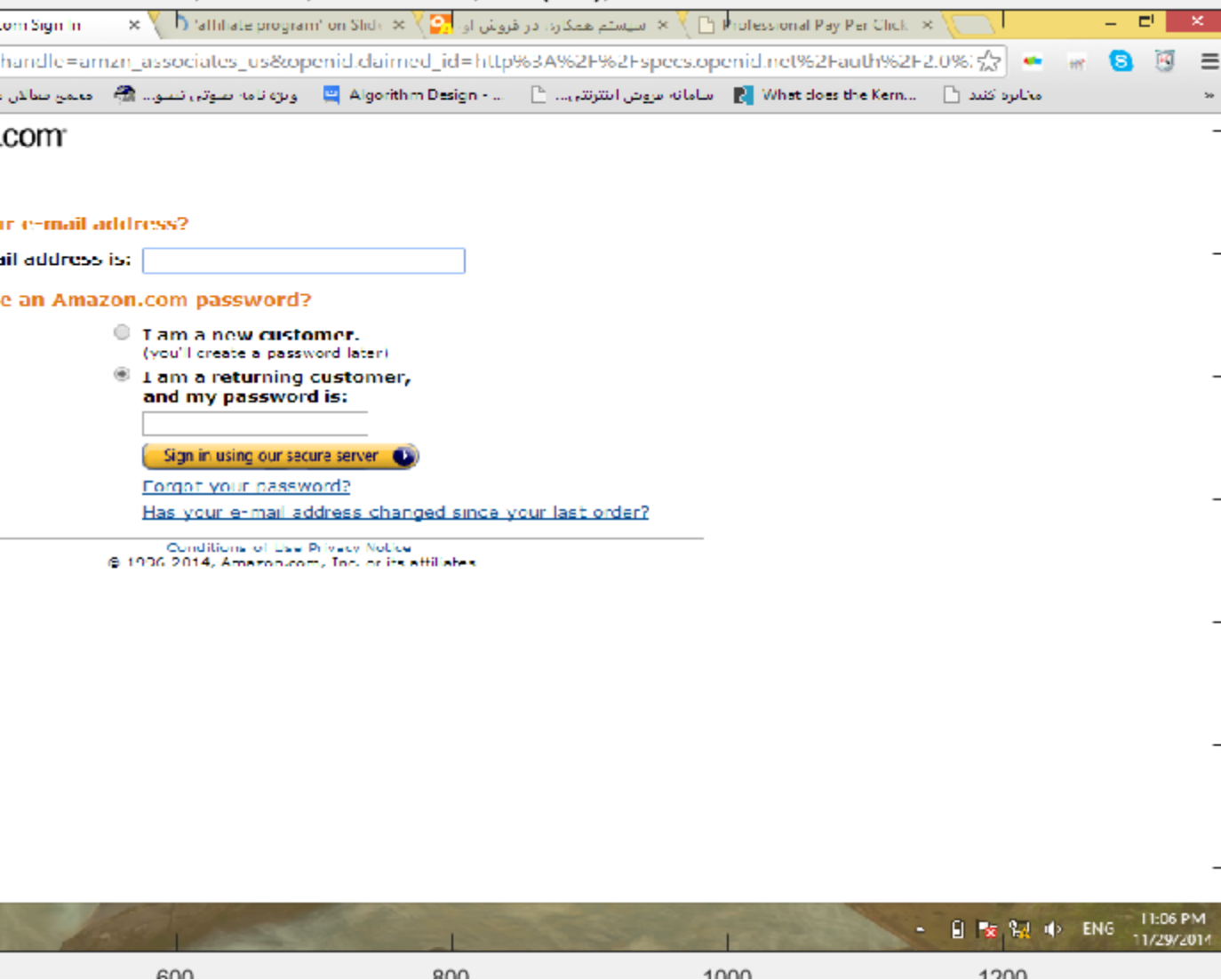
deep-19.mat: web site, website, internet site, site (917), score 0.987



همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۰,۹۸۷)

نتیجه شبکه معمولی:

f.mat: web site, website, internet site, site (917), score 0.949



این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد(امتیاز ۰,۹۴۹)

نتیجه شبکه با ۱۹ لایه:

imagenet-vgg-verydeep-19.mat: Indian cobra, *Naja naja* (64), score 1.000



همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۱)

نتیجه شبکه معمولی:



این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد (امتیاز ۰,۹۶۷).

نتیجه شبکه با ۱۹ لایه:



enet-vgg-verydeep-19.mat: castle (484), score 0.575



همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۰,۷۸۹)

نتیجه شبکه معمولی:

imagenet-vgg-f.mat: castle (484), score 0.909



این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد (امتیاز ۰,۳۶۸)

نتیجه شبکه با ۱۹ لایه:

g-verydeep-19.mat: flagpole, flagstaff (558), score 0.398



همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۰,۷۸۹)

نتیجه شبکه معمولی:



enet-vgg-f.mat: flagpole, flagstaff (558), score 0.759

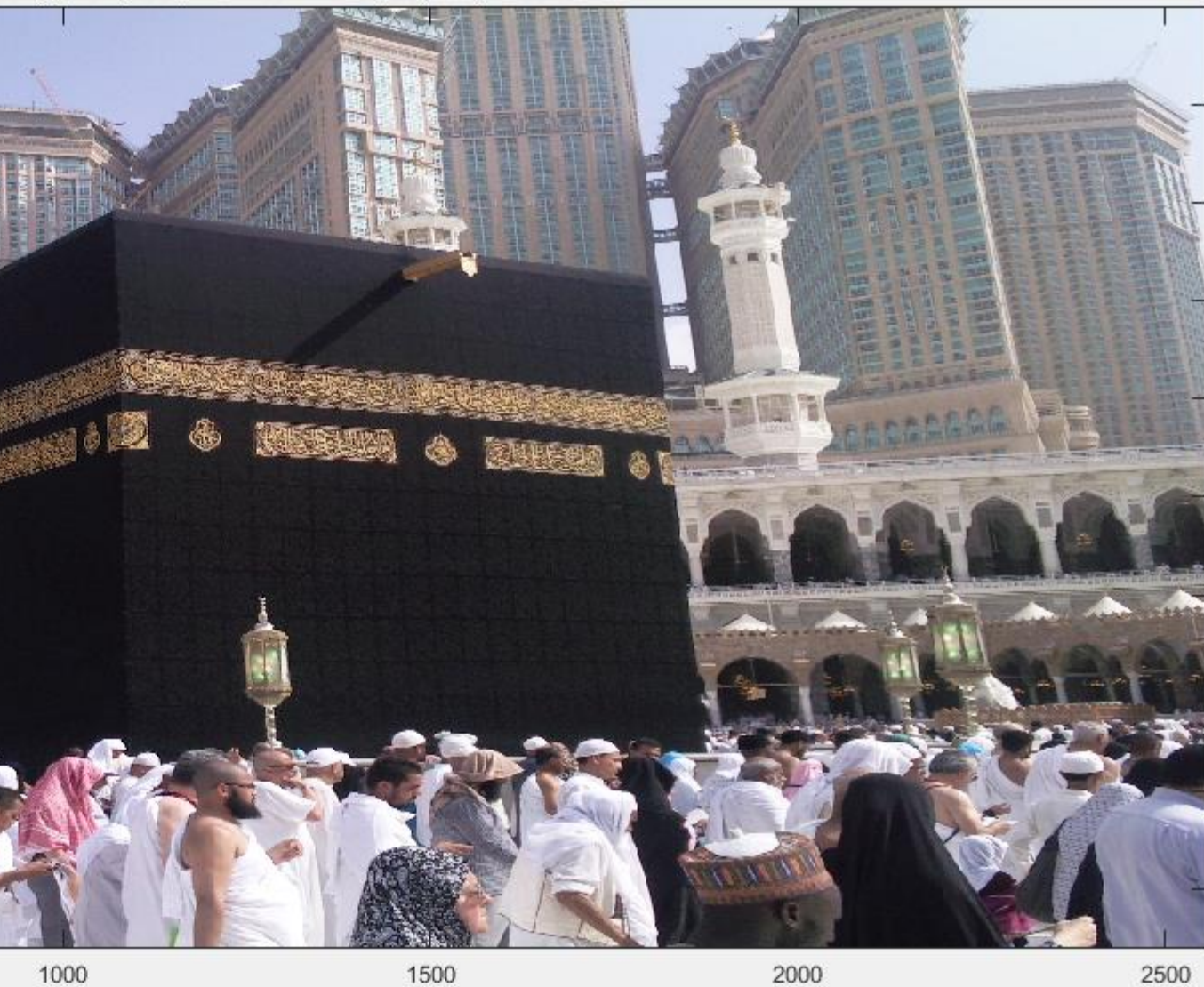


این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد (امتیاز ۰,۳۶۸)

نتیجه شبکه با ۱۹ لایه:

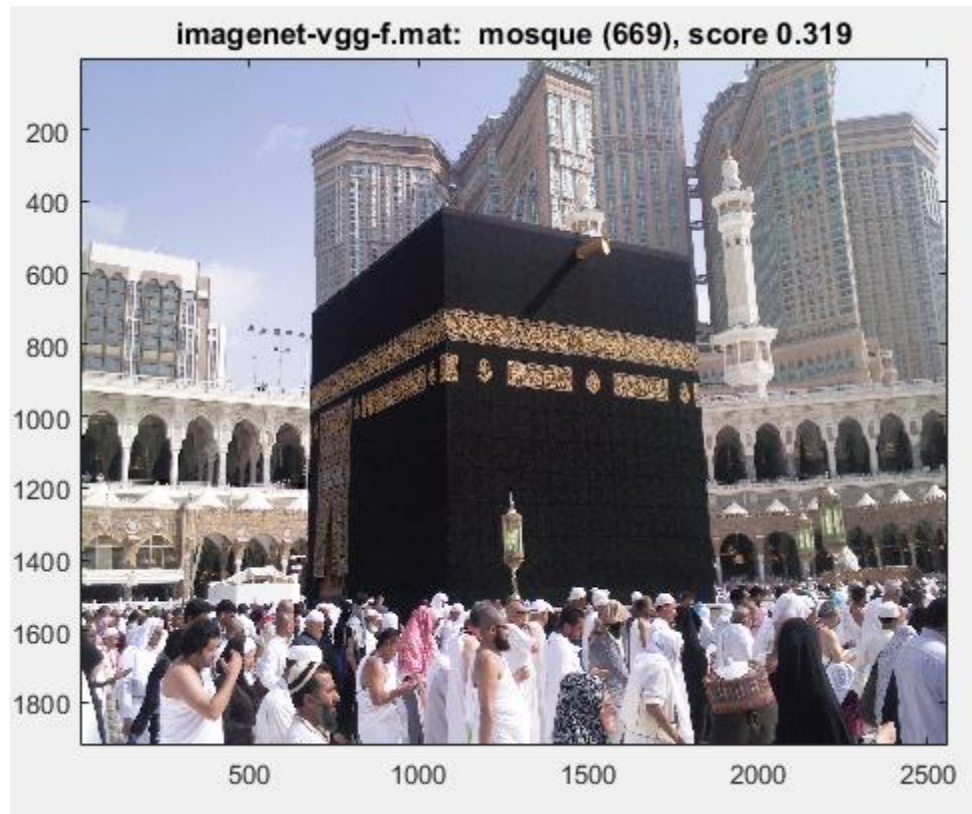


t-vgg-verydeep-19.mat: mosque (669), score 0.331

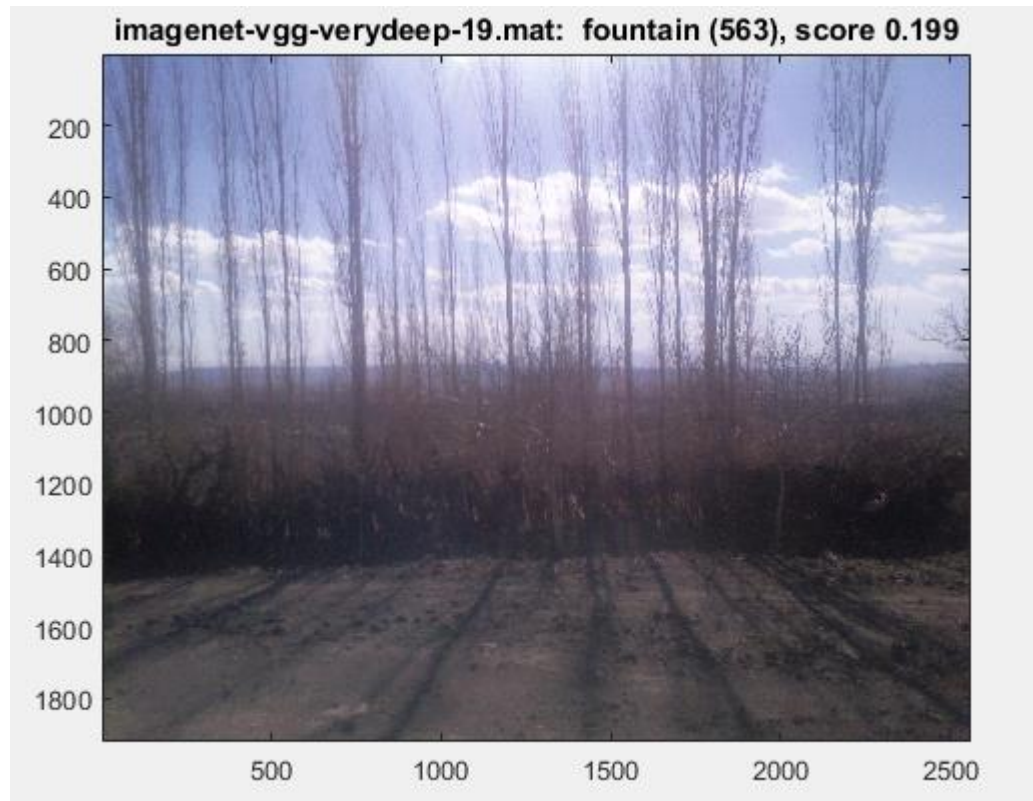


همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۰,۷۸۹)

نتیجه شبکه معمولی:

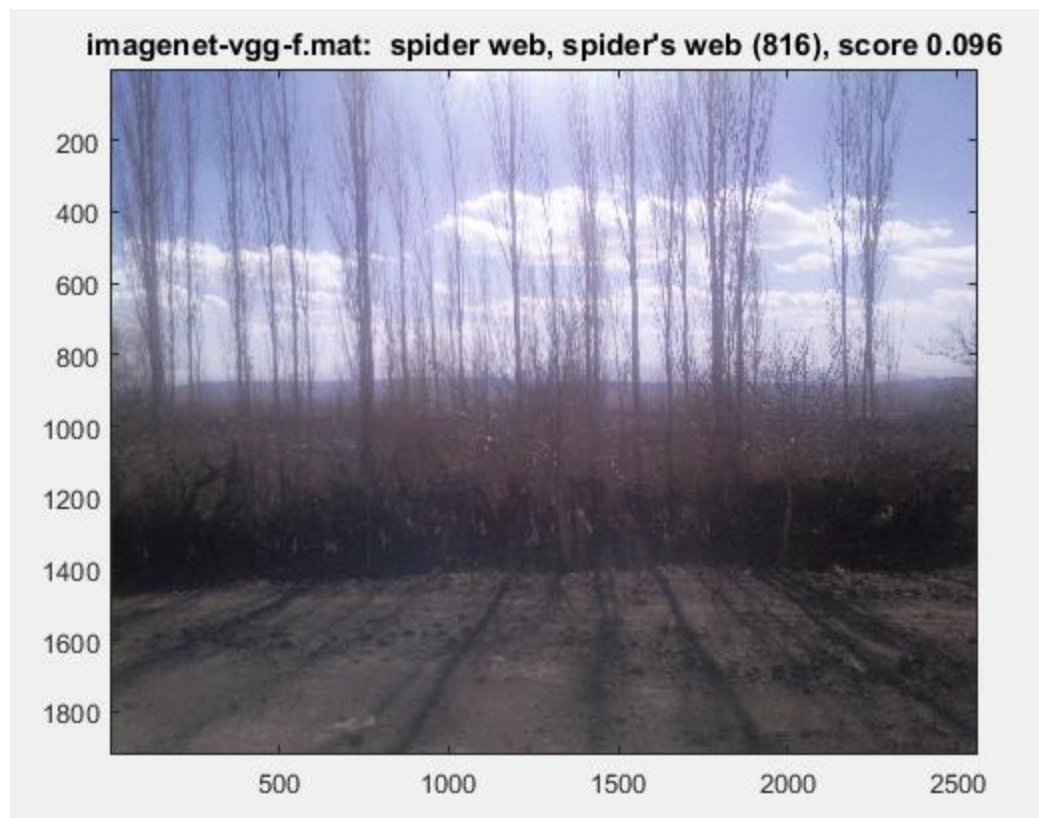


این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد (امتیاز ۰,۳۶۸)  
نتیجه شبکه با ۱۹ لایه:



همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۰,۷۸۹)

نتیجه شبکه معمولی:



این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد (امتیاز ۰,۳۶۸)

نتیجه شبکه با ۱۹ لایه:

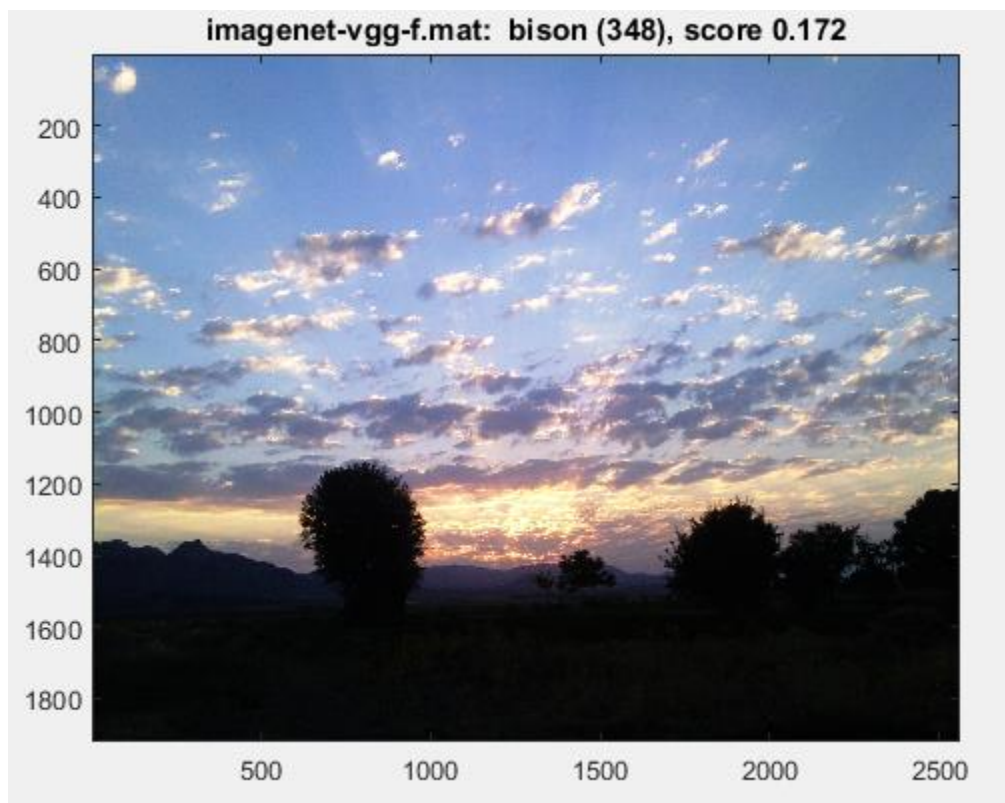


deep-19.mat: radio telescope, radio reflector (756), score 0.275



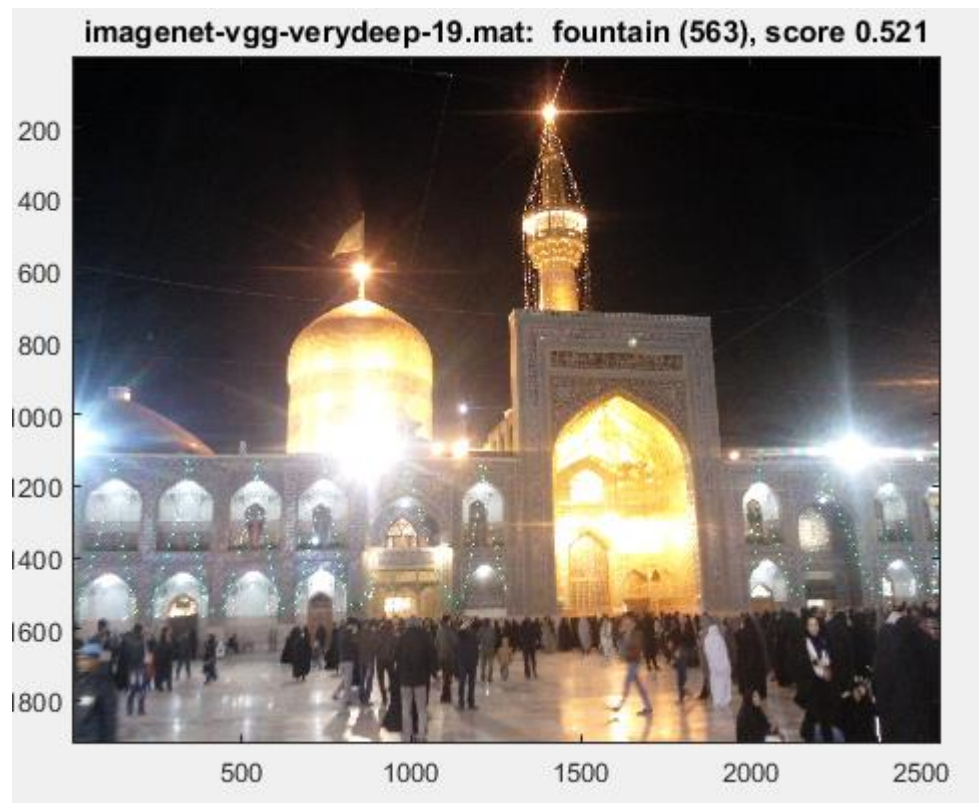
همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۰,۷۸۹)

نتیجه شبکه معمولی:



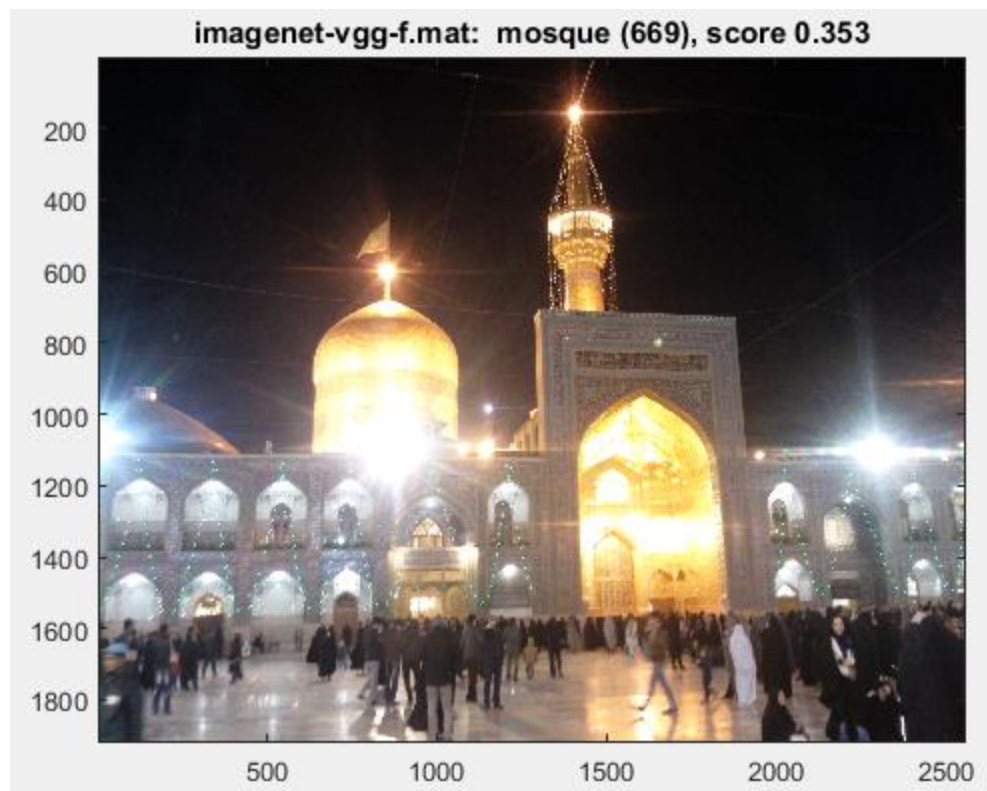
این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد (امتیاز ۰,۳۶۸)

نتیجه شبکه با ۱۹ لایه:



همانطور که مشاهده میکنید این تصویر را با دقت بالایی درست تشخیص داده است.(امتیاز ۰,۷۸۹)

نتیجه شبکه معمولی:



این شبکه هم درست تشخیص داده است اما درصد اطمینان کمتری دارد (امتیاز ۰,۳۶۸)

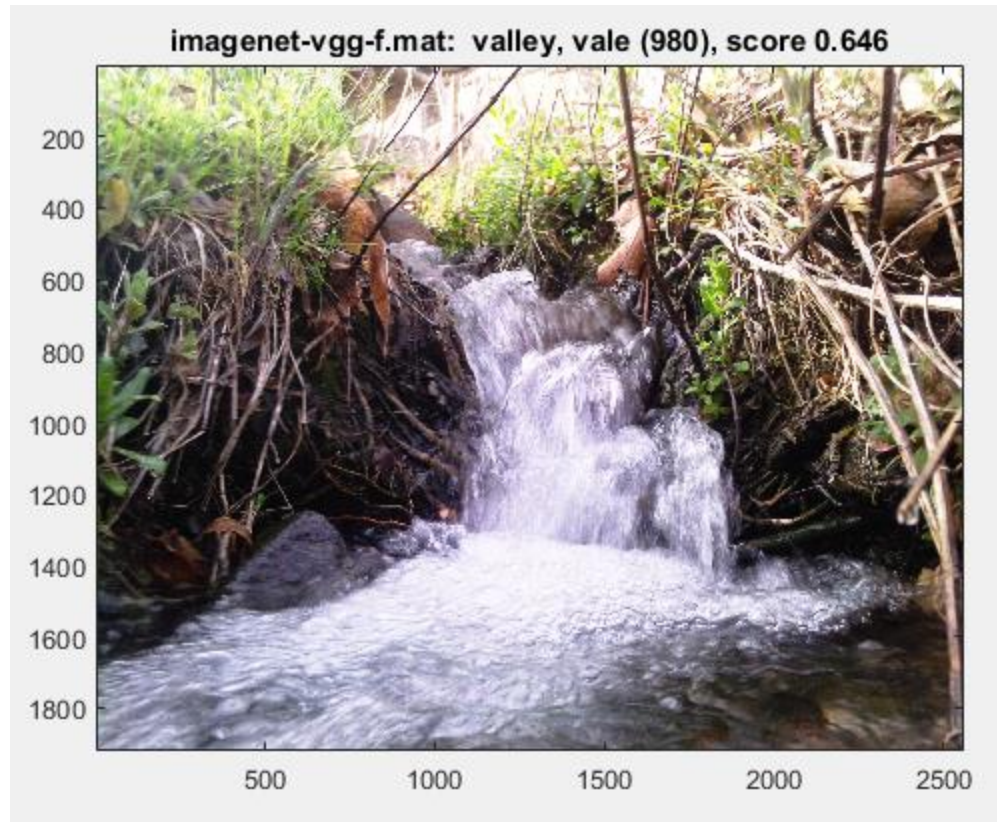
۱۹ لایه:



imagenet-vgg-verydeep-19.mat: valley, vale (980), score 0.213



معمولی:



مد این قسمت به صورت زیر می باشد:

```
clc
clear
close all;
%add pre MatConvNet setup
vl_setupnn;
% load the pre-trained CNN(regular CNN)
net = load('imagenet-vgg-f.mat');

% load and preprocess an image
im = imread('F:\Documents\MATLAB\Neural Network\HW3\Google
Net\Pictures\1 (12).jpg') ;
im_ = single(im) ; % note: 0-255 range
im_ = imresize(im_, net.normalization.imageSize(1:2)) ;
im_ = im_ - net.normalization.averageImage ;
% run the CNN
res = vl_simplenn(net, im_) ;

% show the classification result
```

```

scores = squeeze(gather(res(end).x)) ;
[bestScore, best] = max(scores);
figure(1) ; clf ; imagesc(im) ;
title(sprintf('%s %s (%d), score %.3f',...
'imagenet-vgg-f.mat: ',net.classes.description{best}, best,
bestScore)) ;
%%
clc;
% load the pre-trained CNN with 19 layer(verydeep)
net = load('imagenet-vgg-verydeep-19.mat');

% run the CNN
res = vl_simplenn(net, im_) ;

% show the classification result
scores = squeeze(gather(res(end).x)) ;
[bestScore, best] = max(scores);
figure(2) ; clf ; imagesc(im) ;
title(sprintf('%s %s (%d), score %.3f',...
'imagenet-vgg-verydeep-19.mat:
',net.meta.classes.description{best}, best, bestScore)) ;

```

## سوال ۲-ب

برای این سوال در ابتدا ورودی را خوانده و سپس آن را به صورت تصادفی به ۲ قسمت داده های آموزشی و تست تبدیل میکند و سپس ویژگی های لایه ماقبل آخر شبکه گوگل نت را استخراج میکنیم و سپس ویژگی ها بدست آمده را به یک شبکه عصبی میدهیم و نتایج را بررسی میکنیم، کد این قسمت به صورت زیر خواهد بود:

```

% clc
clear
close all;
vl_setupnn;

% load the pre-trained CNN
cnnModel.net = load('F:\Documents\MATLAB\Neural
Network\HW3\Google Net\Data\imagenet-vgg-f.mat');

%% Load data from folder

```

```
% Use imageSet to load images stored in pet_images folder
% imset = imageSet('F:\Documents\MATLAB\Neural
Network\HW3\Google Net\Data\Airplane Images','recursive');
imset = imageSet('F:\Documents\MATLAB\Neural
Network\HW3\Google Net\Data\Pet Images','recursive');
```

```
% Load and resize images for prediction
counter = 0;
[~,im_size] = size(imset);
for j=1:im_size
    for i = 1:imset(j).Count
        counter = counter +1;
        images{counter} = read(imset(j),i);%real size
        images2(counter).contain=
imresize(read(imset(j),i),[224,224]); %resize size(fixd
size)
    end
end
```

```
%% Get the image labels
Labels = getImageLabels(imset);
summary(Labels) % Display class label distribution
```

```
%% shuffeing images
min_idx=1;
max_idx=counter;
idx = randperm(max_idx-min_idx+1,40);
```

```
% Preallocate arrays with fixed size
[im_hight,im_width,im_depth] = size(images2(1).contain);
images_matrix = zeros(im_hight,im_width,im_depth,counter);
% images_matrix =
zeros([im_hight,im_width,im_depth,counter],'single');
for i=min_idx:max_idx

    myImg = im2double(images2(i).contain);
```

```

%      myImgNorm = featureNormalize(myImg);
% Normalize the Image:
myRange = getrangefromclass(myImg(1));
newMax = myRange(2);
newMin = myRange(1);
myImgNorm = (myImg - min(myImg(:)))*(newMax -
newMin)/(max(myImg(:)) - min(myImg(:))) + newMin;
images_matrix(:, :, :, i) = im2double(myImgNorm);
end

%shuffle test data's
% training images and test images separation
test_images = images_matrix(:, :, :, idx);
images_matrix(:, :, :, idx) = [];
training_images = images_matrix;
% training labels and test labels separation
test_Labels = Labels(idx,1);
Labels(idx,:) = [];
training_Labels= Labels;

%shuffle tarin data's
[label_marix_size,~] = size(training_Labels)
min_idx=1;
max_idx=label_marix_size;
idx = randperm(max_idx-min_idx+1,label_marix_size);

training_images = training_images(:, :, :, idx);
training_Labels = training_Labels(idx,:);
%% Display image and correpondig lable for it
% for i=1:label_marix_size
%         imshow( training_images(:, :, :, i));
%         title(char(training_Labels(i)));
%         pause;
%         close all;
% end

%%

cnnModel.net.layers = cnnModel.net.layers(1:end-1);
cnnModel.info.opts.batchSize = 50;

```

```
cnnFeatures = cnnPredict(cnnModel,training_images);
```

```
training_target = zeros(label_marix_size,1);
```

```
for i=1:label_marix_size
    if strcmpi(char(training_Labels(i)),'0')
        training_target(i,1) = 0;
    else
        training_target(i,1) = 1;
    end
end
```

```
[test_label_marix_size,~] = size(test_Labels);
```

```
test_target = zeros(test_label_marix_size,1);
```

```
for i=1:test_label_marix_size
    if strcmpi(char(training_Labels(i)),'0')
        test_target(i,1) = 0;
    else
        test_target(i,1) = 1;
    end
end
```

```
%% Set inputs for MLP
```

```
t_cnnFeatures = cnnPredict(cnnModel,test_images);
```

```
t_inputs = t_cnnFeatures;
```

```
inputs = cnnFeatures;
```

```
t_feature_target = test_target;
```

```
feature_target = training_target;
```

```
%% Learning MLP with Googl net feature
```

```
google_net = 1;
```

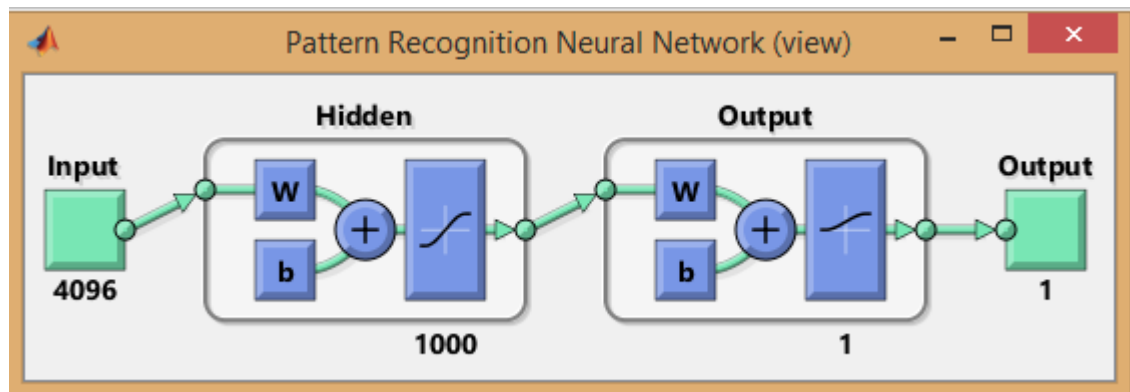
```
MR_MLP;
```

```
%% Train with NN toolbox in matlab if you want!
```

```
% MLP_Inputs= [inputs];
```

```
% Target = feature_target ;
% MLP;
% nptr;
```

متأسفانه دقت بدست خیلی کم بود، من علاوه بر تست روی شبکه عصبی که خودم طراحی کردم از طریق toolbox متلب هم کد را اجرا کردم(کد ها را میتواند در مجموعه کد های پیوستی مشاهده کنید) باز هم نتایج خوب نبودند! نمیدونم چی رو جا انداختم!



دقت:

performance =

0.4414

trainPerformance =

0.3877

valPerformance =

0.3902

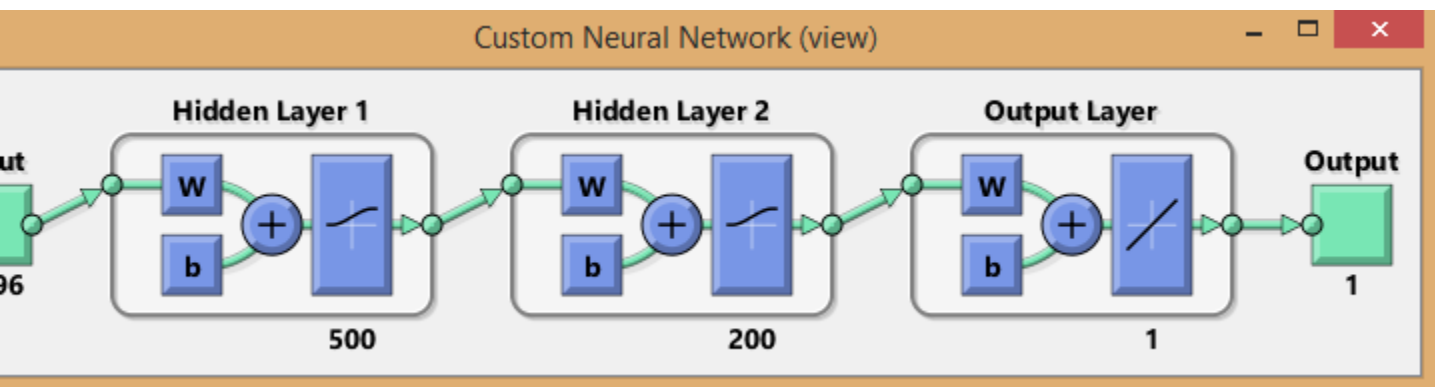
testPerformance =

0.4389

Percentage Correct Classification : 42.500000%

Percentage Incorrect Classification : 57.500000%

شبکه مورد آزمایش بعدی:



دقت:

performance =

16.6744

trainPerformance =



20.5483

valPerformance =

36.0437

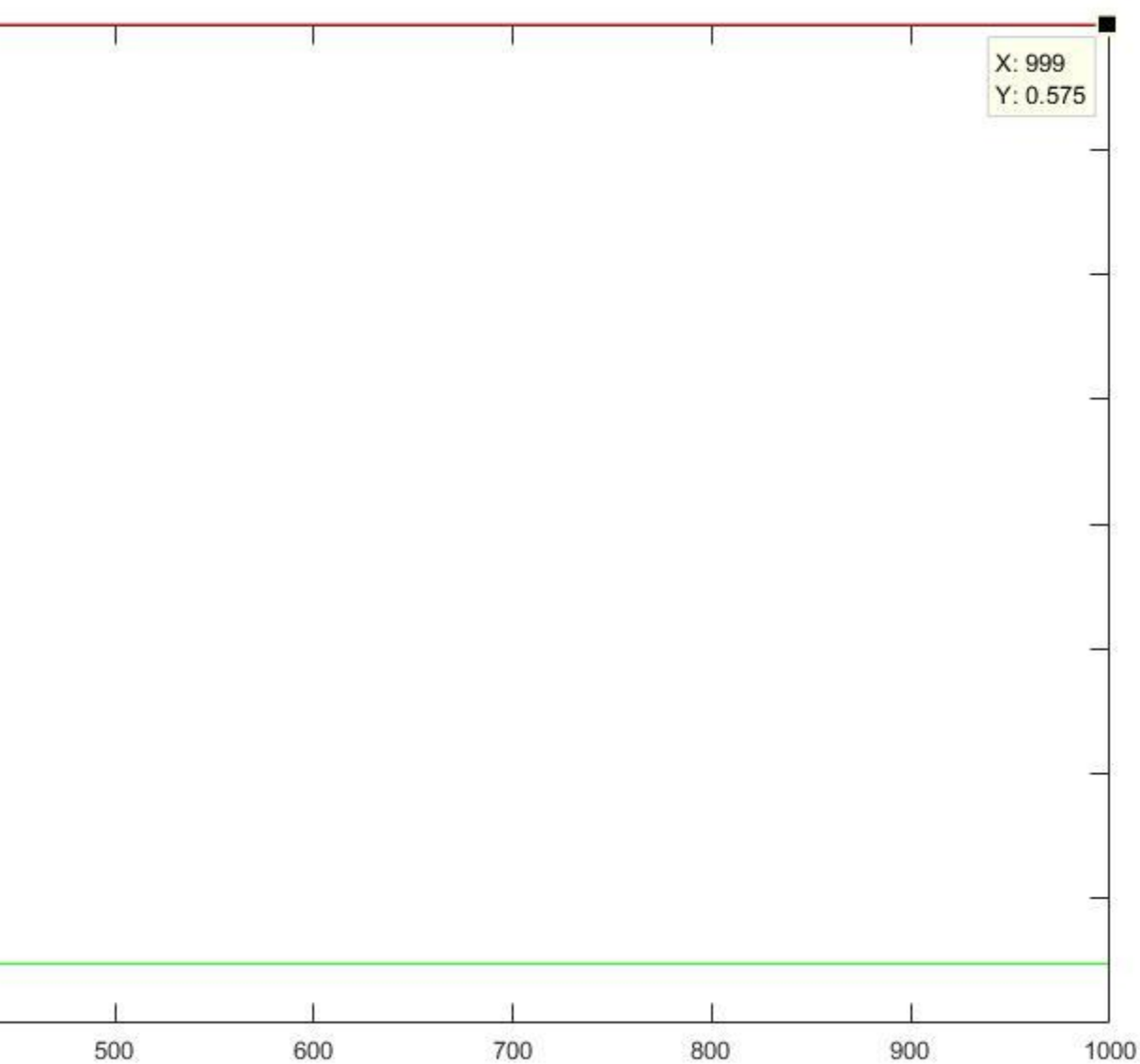
testPerformance =

32.1698

Percentage Correct Classification : 42.500000%

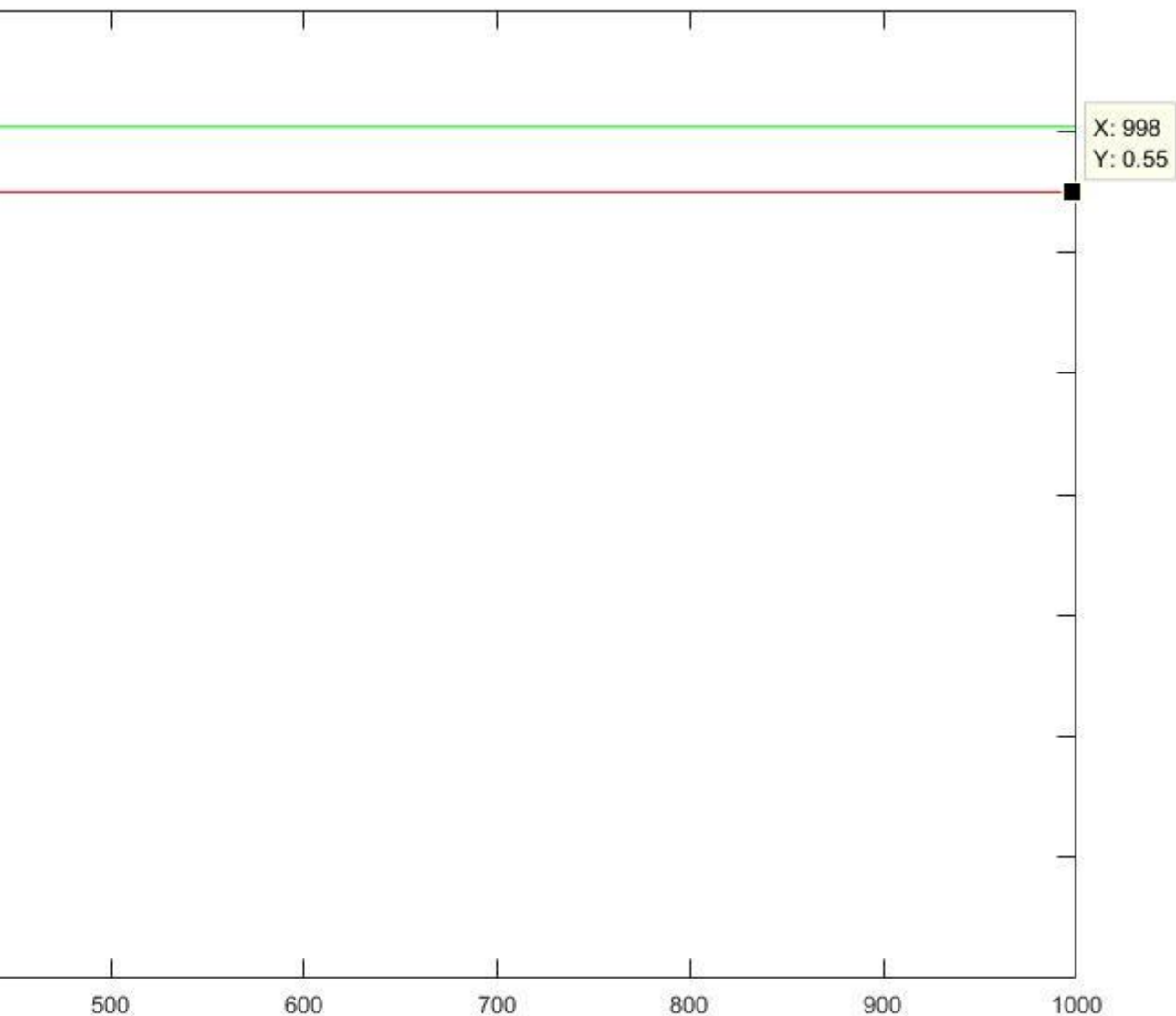
Percentage Incorrect Classification : 57.500000%

و در آخر نتایج شبکه MLP خودم:



دقت روی ۵۷,۵ ثابت مونده و تغییری نکرده است.

نتیجه یک آزمایش دیگر:



من آزمایش های خیلی زیادی رو انجام دادم و معماری ها و پرامترهای زیادی رو تغییر دادم ولی به نتیجه خاصی نرسیدم، به احتمال زیاد ویژگی های استخراج شده توسط من از شبکه گوگل مناسب نیستند که شبکه نمیتواند چیزی یاد بگیرد. اما ویژگی های لایه ها دیگر (بجز لایه آخر) رو هم تست کردم اما به نتیجه خاصی نرسیدم. روی ویژگی های بدست آمده پردازش هایی (مثل نرمال سازی) انجام دادم اما باز هم نتیجه ای نداشت و شبکه ها چیزی رو یاد نمی گرفتند.

## c-2) هدف از این بخش مقایسه روش استخراج ویژگی با الگوریتم PCA به شرحی که در ادامه می آید و سپس طبقه‌بندی تصاویر با یک MLP است.

برای این کار از روش استخراج محورهای eigenface با الگوریتم PCA استفاده نمایید که فایل الگوریتم آن به پیوست ارائه شده است. توضیحات نحوه استفاده در خود فایل تشریح شده است و در ادامه نیز به طور مبسوط توضیح داده می شود. جهت توضیحات تکمیلی و نحوه استخراج ویژگی می‌توانید به آدرس زیر مراجعه نمایید:

مراحل استخراج ویژگی از تصاویر با استفاده از الگوریتم PCA به صورت زیر است:

- ۱- نرمال سازی: تصاویر را خوانده و اندازه‌ی آن‌ها را یک سایز دلخواه نظیر  $32 \times 32$  تغییر دهید. سپس مقادیر پیکسل‌های هر تصویر را در بازه ی [۰-۱] نرمال کنید. در این مرحله فرض اینست که تصاویر شما رنگی نیست. اگر شما از تصاویر رنگی استفاده کرده اید می‌توانید آنها را خاکستری کنید و سپس الگوریتم PCA را اجرا کنید یا یک روش برای استفاده از PCA در تصاویر رنگی پیشنهاد دهید. می‌توانید بدین منظور از سایر کدهای آماده که از PCA بروی تصاویر رنگی استفاده کرده اند بهره ببرید.
- ۲- درصد داده‌ها را به داده ۸۰ جدا کردن تصاویر آموزش و تست: همانطور که در بخش قبل گفته شد، آموزش و ۲۰ درصد را به عنوان داده تست در نظر بگیرید.
- ۳- بردار کردن: هر تصویر  $32 \times 32$  را به یک بردار  $1024 \times 1$  تبدیل کنید.
- ۴- ساختن ماتریس داده‌های آموزشی: تصاویر بردار شده را کنار هم قرار دهید تا ماتریس Xtrn با ابعاد  $1024 \times Ntrn$  به دست آید (تعداد داده‌های آموزشی است).
- ۵- ساختن بردارهای PCA: با استفاده از تابع PCA که به پیوست داده شده است، بردارهای PCA را به دست آورید. تعداد ویژگی‌ها را در متغیر dim قرار دهید.

$$[\text{eigvector}, \text{eigvalue}] = \text{PCA}(\text{Xtrn}, \text{dim});$$

مثلا اگر dim را برابر با ۳۰ قرار دهید، ماتریس eigvector به صورت یک ماتریس  $1024 \times 30$  به دست خواهد آمد (به ابعاد توجه کنید).

- ۶- استخراج ویژگی از تصاویر: برای استخراج ویژگی از رابطه‌ی زیر استفاده کنید:

$$\text{Ytrn} = (\text{Xtrn} * \text{eigvector})';$$

ماتریس  $Y_{trn}$  به صورت  $dim * N_{trn}$  به دست خواهد آمد که بیان کننده این است که از هر تصویر  $dim$  ویژگی استخراج شده است. یعنی هر ستون این ماتریس، بردار ویژگی استخراج شده برای یکی از تصاویر مجموعه‌ی آموزشی می‌باشد.

۷- استخراج ویژگی از داده‌های تست: برای استخراج ویژگی از تصاویر تست کفایت پس از آماده سازی ماتریس  $X_{tst}$ ، مشابه مرحله ۶ عمل کرده و ماتریس  $Y_{tst}$  را به صورت زیر استخراج نمایید:  $Y_{tst} = (X_{tst} * eigvector)^T$ ؛ دقت کنید که در این مرحله نیازی به استفاده از الگوریتم PCA نیست و از همان ماتریس  $eigvector$  که در مرحله‌ی ۵ به دست آمده استفاده کنید.

۸-

برای حل این سوال با توجه به کم بودن داده‌های آموزشی که مشکل  $over\ train$  را در داده‌های آموزشی بوجود می‌آورد، در انتظار نتیجه چندان خوبی نیستیم، اما برای اینکه کارکرد PCA مشخص شود، در ابتدا چند تصویر بازسازی شده را میبینیم، شبکه مورد استفاده ما شبکه چند لایه عصبی تمرین دوم درس شبکه‌های عصبی هست که این شبکه بروز شده و قابلیت‌های بیشتری رو نسبت به شبکه اوایه دارا می‌باشد (در این شبکه قبل از اجرای شبکه پارامترها را از کاربر دریافت میکند و کاربر میتواند روش یادگیری، تابع فعال ساز، اندازه دسته در روش  $mini\ batch$  و نرخ یادگیری و مومنتوم و منتظم سازی و تعداد لایه‌ها و غیره را تعیین کند و از  $adaptive\ learning\ rate$  هم استفاده میکند) در تصویر زیر دیالوگ ورودی را می‌بینید:

Input

Method:(1 for Batch method - 0 for Online method)  
1

Learning rate:  
0.3

alfa:(Momentum coefficient)  
0

lambda:(Regularization coefficient)  
0

Validation check  
100

Iteration  
100

Number of samples  
1000

Number of layers  
3

Batch size  
100

Train(1) or Train2(2) or Train3(3) or Train4(4)  
1

tanh or sigmoid(1 for tanh -2 for sigmoid)  
1

OK Cancel

و حالا نوبت به استفاده از PCA برای استخراج ویژگی می‌رسد:

برای این سوال هم کد PCA را برای تصاویر رنگی نوشتیم و هم سیاه و سفید، اما در نهایت از کد رنگی برای استخراج ویژگی استفاده کردیم، چرا که نتایج بدست مده با این کد بهتر بودند، در ادامه کد استخراج ویژگی PCA را بر روی تصاویر رنگی مشاهده میکنید:

کلیت کار:

ابتدا کانل های عکس رو استخراج میکند و سپس PCA را بروی روی هر کانال (قرمز، سبز، آبی) به صورت مجزا اجرا میکند و سپس هر کانال را کاهش بعد داده (ویژگی ها را استخراج کرده) و سپس کانال های کاهش بعد داده شده را به عنوان ورودی شبکه عصبی استفاده میکند:

چند تصویر بازسازی شده با ۲۵ ویژگی استخراجی:  
تصاویر مجموعه هواپیما ها:



**Original**



**Recovered**



**Original**



**Recovered**



**Original**



**Recovered**





Original



Recovered



Original



Recovered



Original



Recovered



با حفظ ۵ ویژگی:

Original



Recovered



Original



Recovered



Original



Recovered



Original



Recovered



Original



Recovered



Original



Recovered



**Original**



**Recovered**



**Original**



**Recovered**



تصاویر سگ و گربه:

**Original**



**Recovered**





**Original**



**Recovered**



**Original**



**Recovered**



**Original**

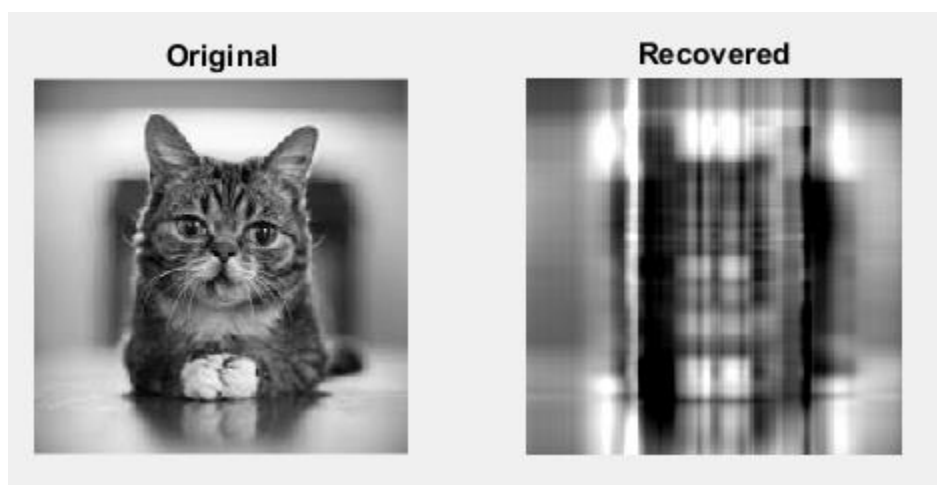
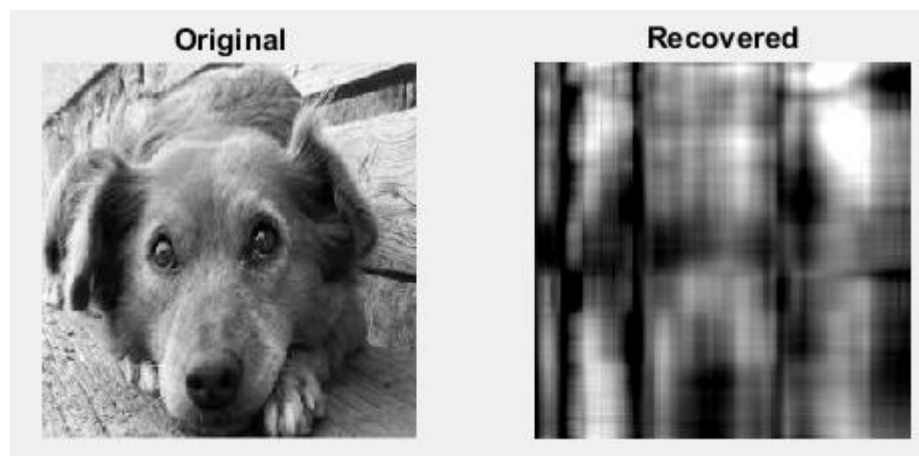


**Recovered**

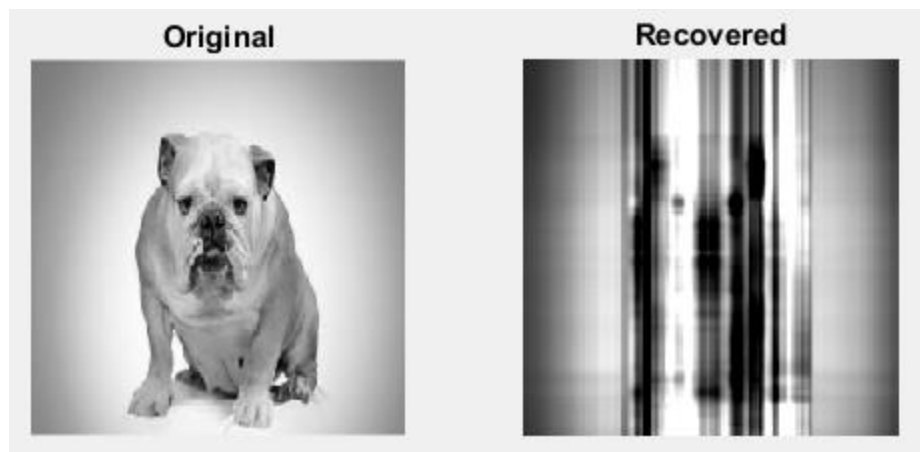
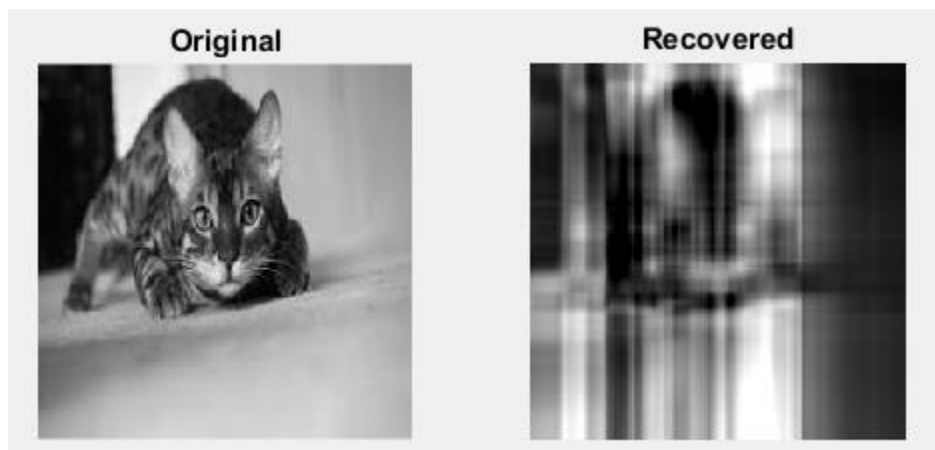
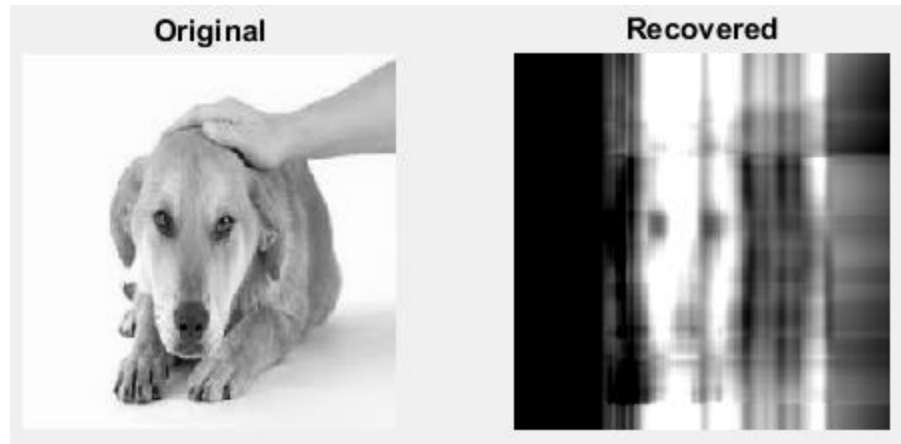




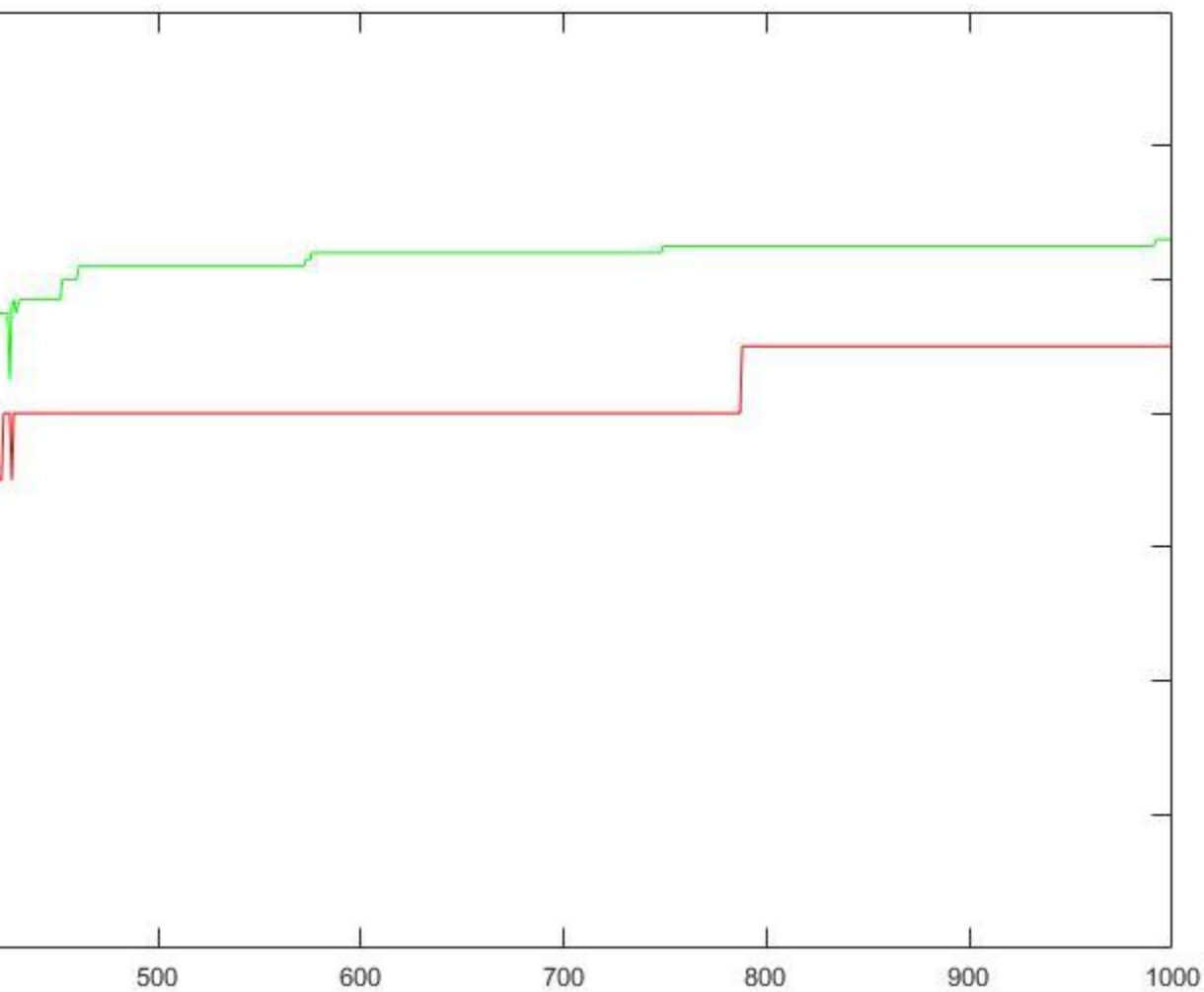
برای اینکه به تفاوت روش های PCA تصاویر رنگی و سیاه و سفید، در ادامه چند تصویر بازسازی شده از کد سیاه و سفید را مشاهده میکنید:(تمامی پارامتر ها در هر ۲ کد یکسان است)





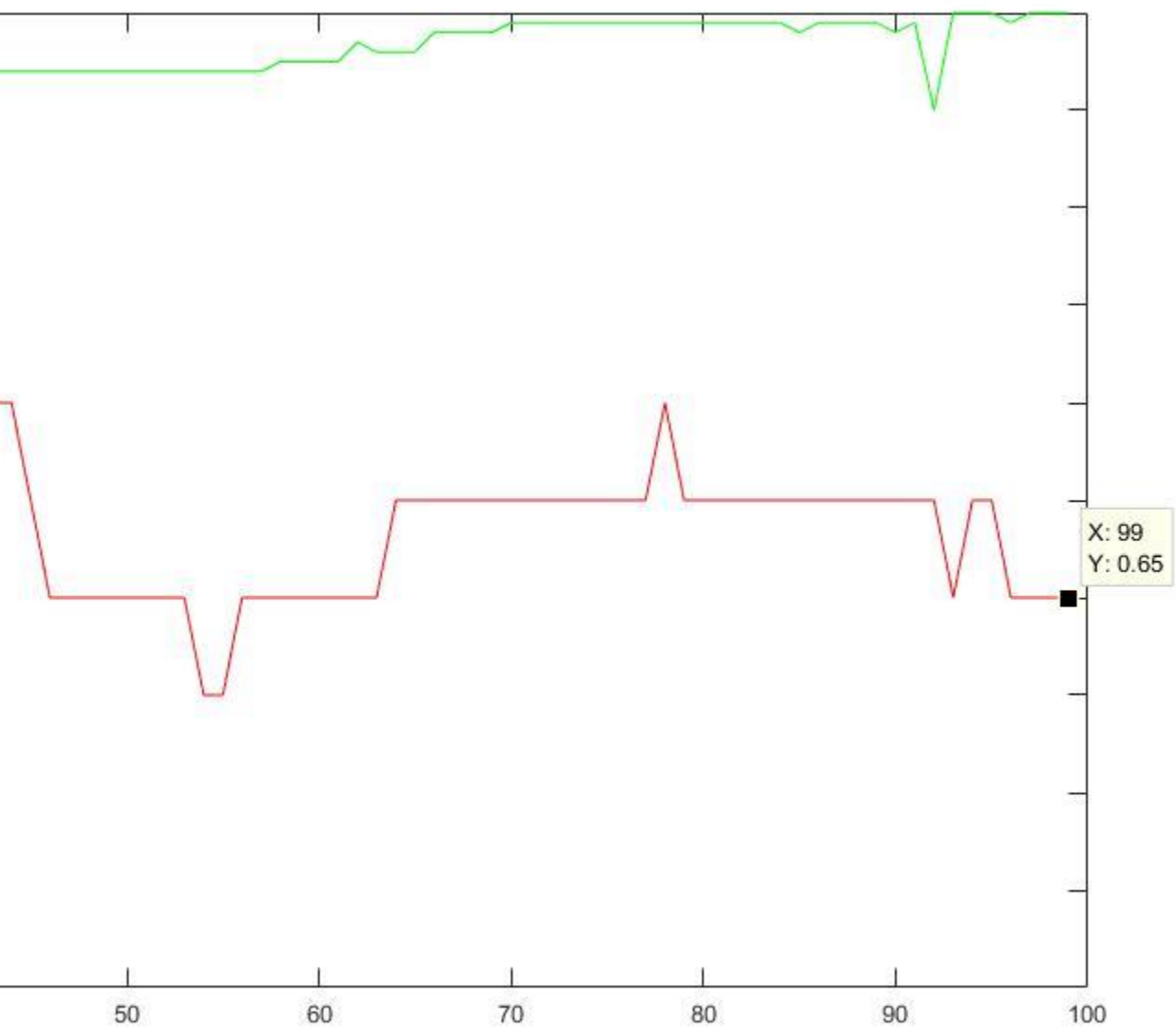


همانطور که مشاهده میکنید، نتایج بازسازی PCA رنگی خیلی بهتر از PCA سیاه و سفید می باشد. بعد از استخراج ویژگی حالا نوبت به آموزش شبکه MLP با این ویژگی ها می باشد، که نتایج آن به صورت زیر می باشد: دقت بر روی نمونه های آموزشی و تست (۱۰۰۰ تکرار الگوریتم یادگیری):



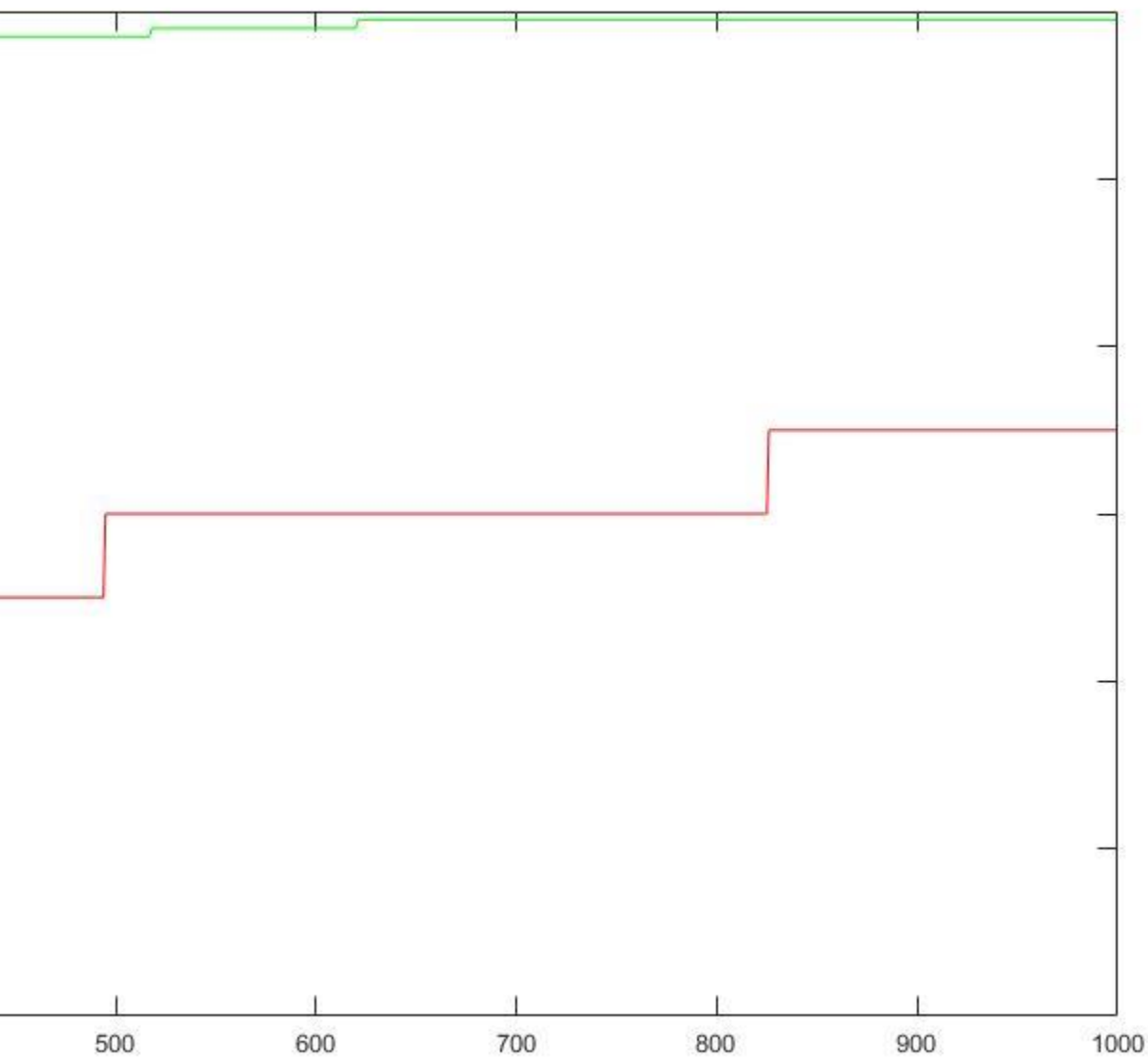
خط سبز دقت بر روی نمونه آموزشی است و خط قرمز دقت بر روی نمونه هاس تست. که دقت تست ۷۵ درصد می باشد.

آزمایش دوم: (۱۰۰ تکرار)



دقت بدست آمده رو نمونه های تست برابر با ۶۵ درصد می باشد.

آزمایش سوم: (۱۰۰۰ تکرار)



و مجددا مشاهده میکنید که به دقت ۷۵ درصد بر روی نمونه های تست رسیدیم.

PCa بر روی تصاویر سیاه و سفید استفاده کردیم و کد آن به صورت زیر می باشد:

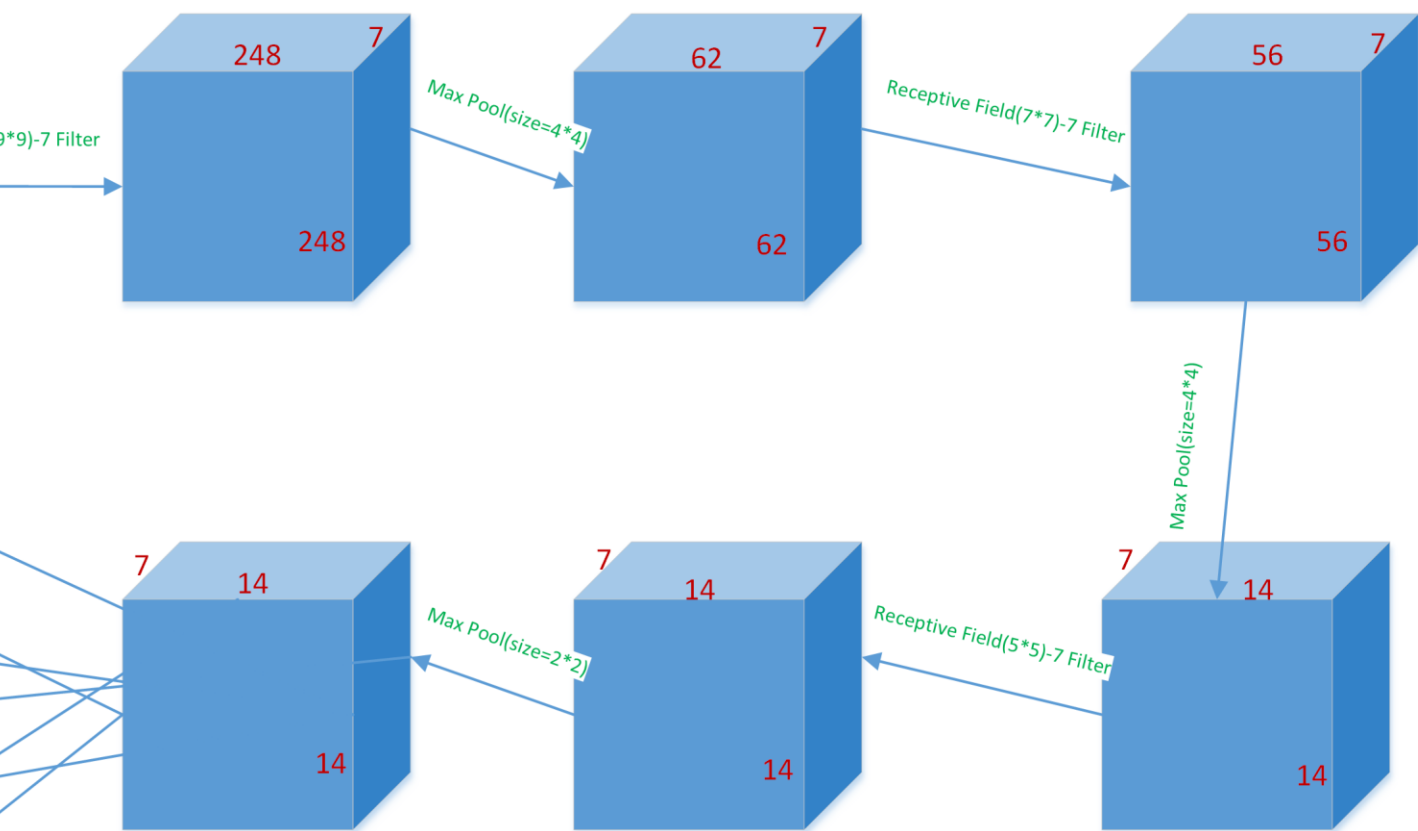
```
%% Train with NN toolbox in matlab if you want!
% MLP_Inputs= [inputs];
% Target = feature_target ;
% % MLP;
```

% nptr;

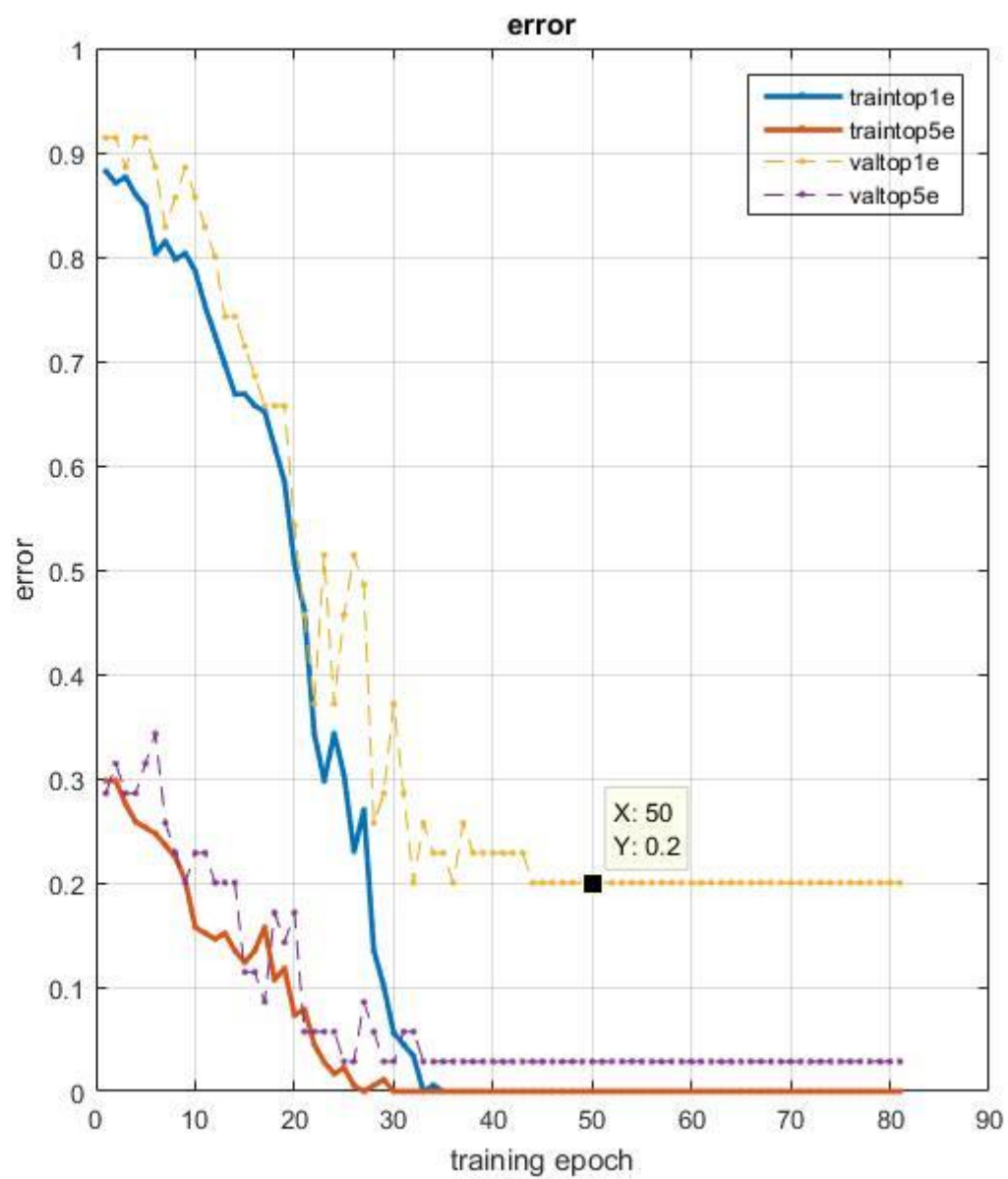
البته من برای اطمینان بیشتر ۲ شبکه MLP را با toolbox متلب اجرا کردم، اما نتایج مثل روش قبل بود.

## سوال ۳:

برای این سوال شبکه زیر را طراحی کردیم:

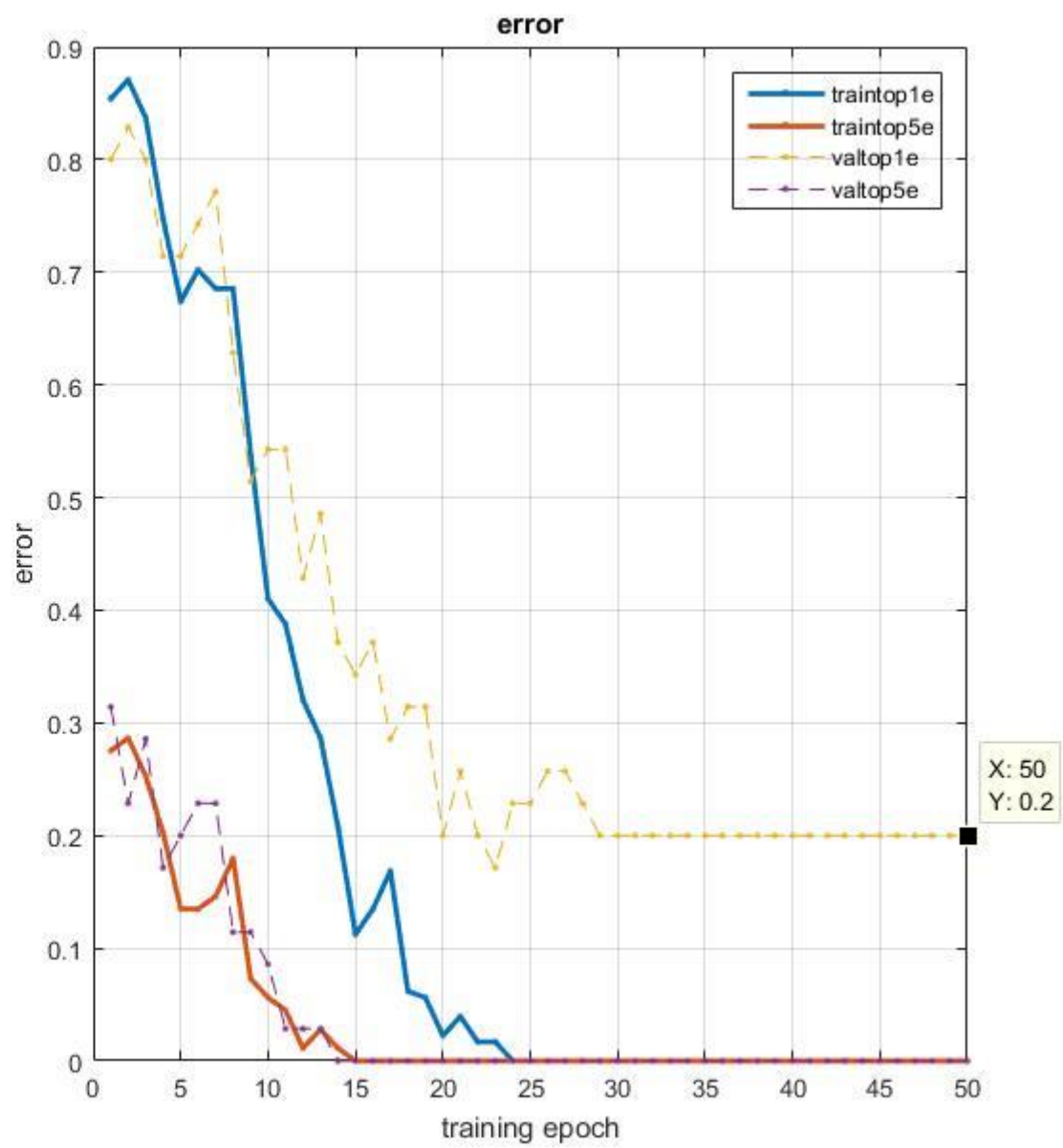


با این شبکه به دقت ۸۰ درصد بر روی نمونه های تست دست یافتیم:

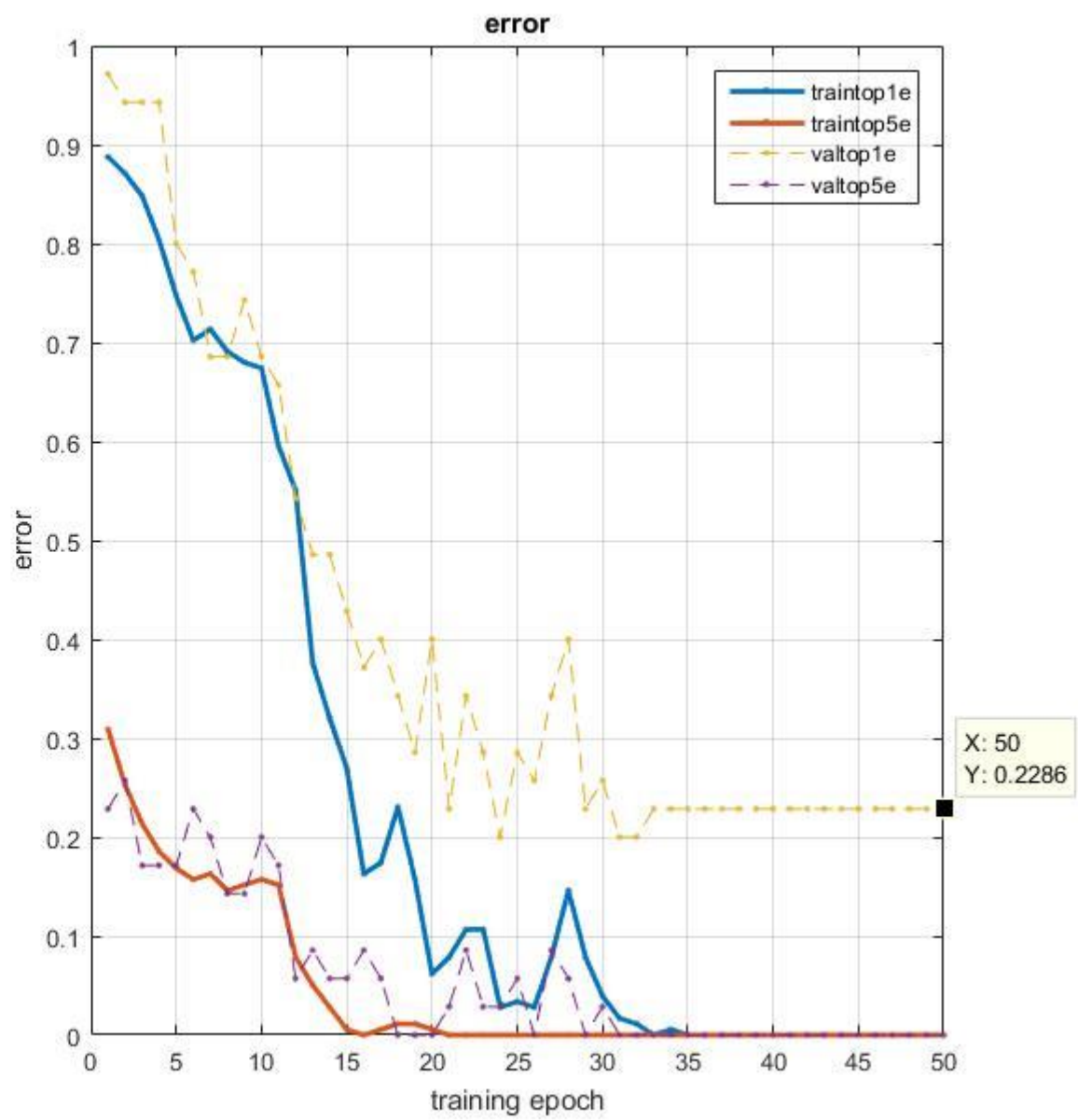


آزمایش دوم (تعداد تکرار ها رو کاهش دادم)

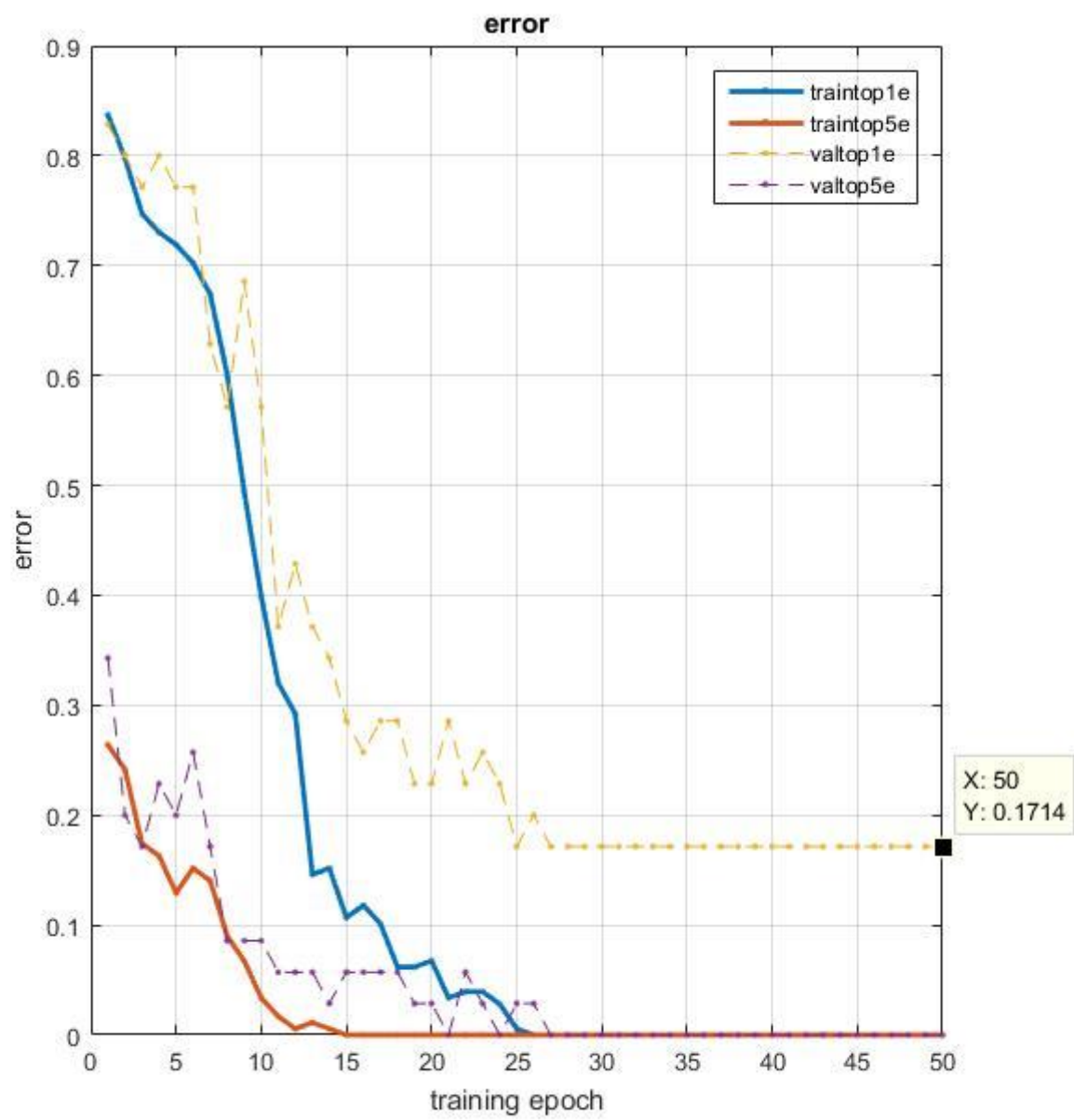




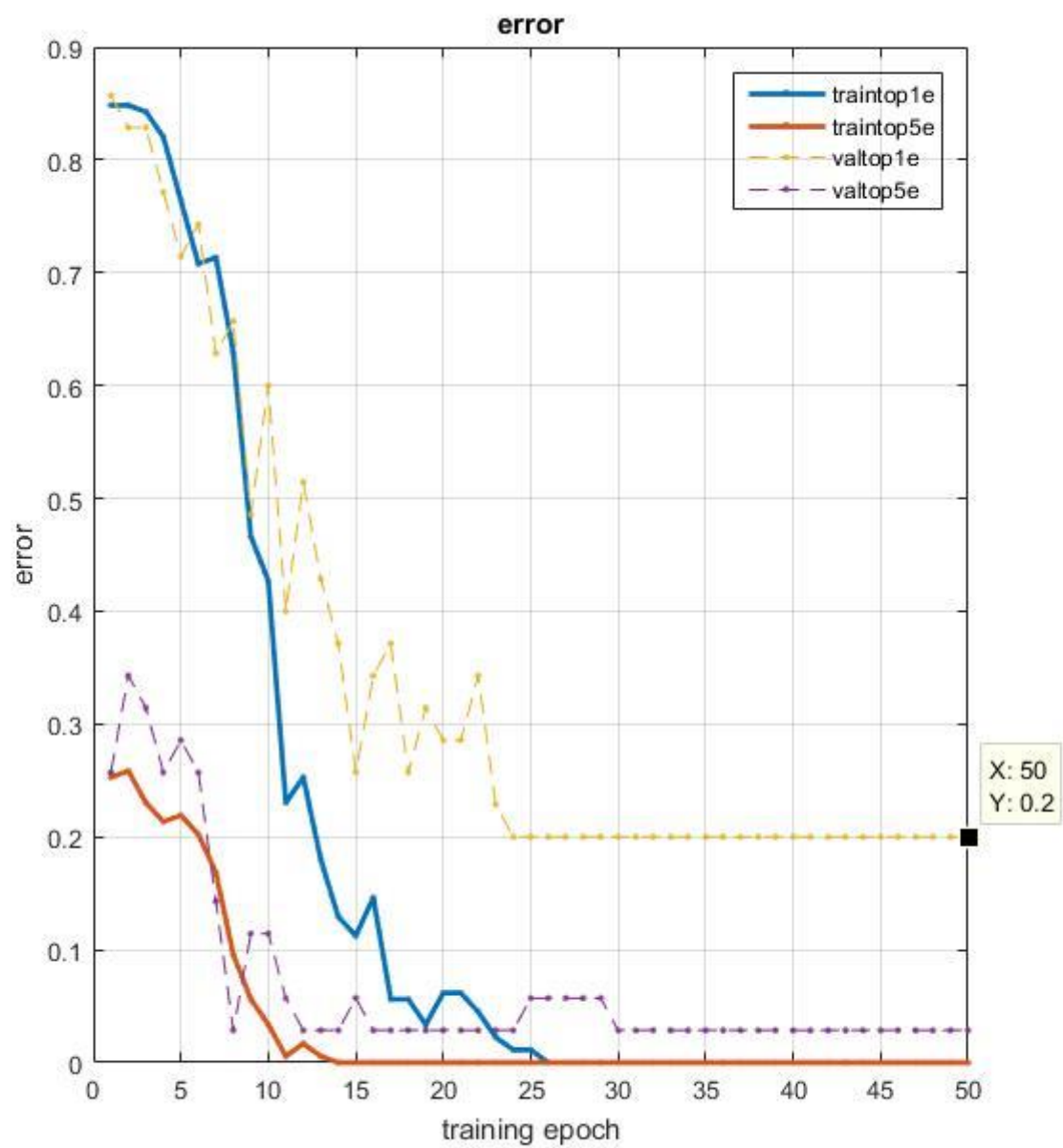
آزمایش سوم:



آزمایش چهارم:



آزمایش پنجم:



میانگین دقت بر روی نمونه های تست:

۸۰ درصد

کد شبکه طراحی شده به صورت زیر می باشد:

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

```

%add pre MatConvNet setup
vl_setupnn;
%% Set some network parameters
opts.train.batchSize = 20;
opts.train.numEpochs = 50 ;
opts.train.learningRate = 0.001 ;

%% Loda Data and normalization
imdb = getJaffeImdb('F:\Documents\MATLAB\Neural
Network\HW3\Jaffe\Data\jaffe train',...
    'F:\Documents\MATLAB\Neural Network\HW3\Jaffe\Data\jaffe
test',[256 256 1]);
% Image normalization
imageMean = mean(imdb.images.data(:)) ;
imdb.images.data = imdb.images.data - imageMean ;
Factor=0.01 ;
%% Network architecture
net.layers = {} ;
%           Receptive Field(9*9)-7 Filter
net.layers{end+1} = struct('type', 'conv', ...
    'weights',
    {{Factor*randn(9,9,1,7, 'single'),...
        zeros(1, 7, 'single')}}}, ...
    'stride', 1, ...
    'pad', 0) ;
% Max Pool(size=4*4)
net.layers{end+1} = struct('type',
'pool', ...
    'method', 'max', ...
    'pool', [4 4], ...
    'stride', 4, ...
    'pad', 0) ;
% Receptive Field(7*7)-7 Filter
net.layers{end+1} = struct('type',
'conv', ...
    'weights',
    {{Factor*randn(7,7,7,7, 'single'),...
        zeros(1, 7, 'single')}}}, ...
    'stride', 1, ...

```

```

        'pad', 0) ;
% Max Pool(size=4*4)
net.layers{end+1} = struct('type',
'pool', ...
    'method', 'max', ...
    'pool', [4 4], ...
    'stride', 4, ...
    'pad', 0) ;
% Receptive Field(5*5)-7 Filter
net.layers{end+1} = struct('type',
'conv', ...
    'weights',
{{Factor*randn(5,5,7,7, 'single'),...
zeros(1, 7, 'single')}}}, ...
    'stride', 1, ...
    'pad', 0) ;
% Max Pool(size=2*2)
net.layers{end+1} = struct('type',
'pool', ...
    'method', 'max', ...
    'pool', [2 2], ...
    'stride', 2, ...
    'pad', 0) ;
% Receptive Field(5*5)-7 Filter as
Fully connected layer
net.layers{end+1} = struct('type',
'conv', ...
    'weights',
{{Factor*randn(5,5,7,7, 'single'),...
zeros(1, 7, 'single')}}}, ...
    'stride', 1, ...
    'pad', 0) ;
% activation function
net.layers{end+1} = struct('type', 'softmaxloss') ;

%% Train yhe network
[net, info] = cnn_train(net, imdb,
@getBatch,opts.train,'val', find(imdb.images.set == 3)) ;

```

البته برای تست معماری های مختلفی را تست کردم و پارامتره را نیز تغییر دادم و همچنین از تکنیک drop out استفاده کردم، اما نتیجه بهتری حاصل نشد.

برخی از پارامترهای تغییر یافته و لایه drop out :

```
opts.train.batchSize = 20;
opts.numSubBatches = 2 ;
opts.train.numEpochs = 100 ;
opts.train.learningRate = 0.001 ;

opts.backPropDepth = +inf ;
opts.cudnn = true ;
opts.weightDecay = 0.0005 ;
opts.momentum = 0.9 ;
opts.errorFunction = 'multiclass' ;

net.layers{end+1} = struct('type', 'dropout', 'name', ...
                          'dropout2', 'rate', 0.3);
```



