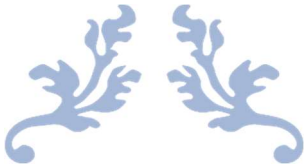


CONVOLUTED NEURAL NETWORKS USING MATLAB

Muhammad Zaid Kamil

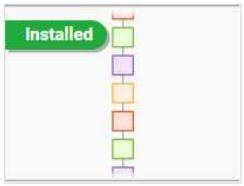
Installation:

Step 1: Install MathWorks 2019

Step 2: Add ons: Deep Learning Toolkit

Step3: Install Deep Learning Toolbox Model

Packages Installed:



Deep Learning Toolbox Model for AlexNet Network
by MathWorks Deep Learning Toolbox Team STAFF

Pretrained AlexNet network model for image classification

MathWorks Optional Feature

MATLAB Support Package for USB Webcams
by MathWorks Image Acquisition Toolbox Team STAFF

Acquire images and video from UVC compliant webcams.

Hardware Support

Overview | Reviews (155) | Discussions (190)

```
>> net=alexnet

net =

  SeriesNetwork with properties:

         Layers: [25×1 nnet.cnn.layer.Layer]
     InputNames: {'data'}
    OutputNames: {'output'}
```

25 layers of different architecture" 5 Convolution layer. Layers of Alexnet: showcase of 25 layers

```
25×1 Layer array with layers:

   1   'data'     Image Input                  227×227×3 images with 'zerocenter' normalization
   2   'conv1'    2-D Convolution              96 11×11×3 convolutions with stride [4  4] and paddinç
   3   'relu1'    ReLU                         ReLU
   4   'norm1'    Cross Channel Normalization  cross channel normalization with 5 channels per elemer
   5   'pool1'    2-D Max Pooling              3×3 max pooling with stride [2  2] and padding [0  0
   6   'conv2'    2-D Grouped Convolution      2 groups of 128 5×5×48 convolutions with stride [1  1]
   7   'relu2'    ReLU                         ReLU
   8   'norm2'    Cross Channel Normalization  cross channel normalization with 5 channels per elemer
   9   'pool2'    2-D Max Pooling              3×3 max pooling with stride [2  2] and padding [0  0
  10   'conv3'    2-D Convolution              384 3×3×256 convolutions with stride [1  1] and paddir
  11   'relu3'    ReLU                         ReLU
  12   'conv4'    2-D Grouped Convolution      2 groups of 192 3×3×192 convolutions with stride [1  1
  13   'relu4'    ReLU                         ReLU
  14   'conv5'    2-D Grouped Convolution      2 groups of 128 3×3×192 convolutions with stride [1  1
  15   'relu5'    ReLU                         ReLU
  16   'pool5'    2-D Max Pooling              3×3 max pooling with stride [2  2] and padding [0  0
  17   'fc6'      Fully Connected              4096 fully connected layer
  18   'relu6'    ReLU                         ReLU
  19   'drop6'    Dropout                      50% dropout
  20   'fc7'      Fully Connected              4096 fully connected layer
  21   'relu7'    ReLU                         ReLU
  22   'drop7'    Dropout                      50% dropout
  23   'fc8'      Fully Connected              1000 fully connected layer
  24   'prob'     Softmax                      softmax
  25   'output'   Classification Output        crossentropyex with 'tench' and 999 other classes
```

Matlab Code

```
% Objection Detection using CNN -
outputFolder = fullfile('C:\Training\');%Returns the path of the dataset
rootFolder = fullfile(outputFolder, '101_ObjectCategories');

% categories_new = {'faces','Orange','watch','cup','banana'};
categories_new = {'airplanes','dollar_bill','watch', 'faces','banana'};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Preprocessing - setting all the labels to be equal to min number of
% images
imds = imageDatastore(fullfile(rootFolder,categories_new),'LabelSource','foldernames');
tbl = countEachLabel(imds);
minSetCount = min(tbl{:,2});% Taking the minimum value of the objects
imds = splitEachLabel(imds, minSetCount, 'randomized');
countEachLabel(imds)

% Training images = 70%, Testing Images = 30%
[imdsTrain,imdsVal] = splitEachLabel(imds,0.7,'randomized');
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,9);
figure
for i = 1:9
    subplot(3,3,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end

alex_net = alexnet;
imageSize = alex_net.Layers(1).InputSize; % [227 227 3]
```

File Path

Categories of Classes

Including subfolders

Using AlexNet and Image size 227x227(HXW)

```
layersTransfer = alex_net.Layers(1:end-3);


numClasses = numel(categories(imdsTrain.Labels))
layers = [
    layersTransfer
    fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];


augmentedTrainingSet = augmentedImageDatastore(imageSize, ...
    imdsTrain, 'ColorPreprocessing', 'gray2rgb');

augmentedValSet = augmentedImageDatastore(imageSize, ...
    imdsVal, 'ColorPreprocessing', 'gray2rgb');
```

```
% Here we train the network
netTransfer = trainNetwork(augmentedTrainingSet,layers,options)

% Classify Validation Images
[YPred,scores] = classify(netTransfer,augmentedValSet);

% Showing the results
idx = randperm(numel(imdsVal.Files),4);
figure
for i = 1:4
    subplot(2,2,i)
    I = readimage(imdsVal,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label));
end
```

```
%Testing my face
Path = 'C:\MATLAB';
image_name = 'zaid-picture.jpeg';
test_predict = imread(fullfile(Path,image_name));
ds = augmentedImageDatastore(imageSize, ...
    test_predict, 'ColorPreprocessing', 'gray2rgb');
[YPred_new,scores_new] = classify(netTransfer,ds);
%I_read = imread(test_predict);
label = YPred_new;
figure
imshow(test_predict)
title({char(label)})
%title({char(label),num2str(max(scores_new),2)})

predictedLabels = classify(netTransfer, augmentedValSet);
accuracy = mean(predictedLabels == imdsVal.Labels)

plotconfusion(imdsVal.Labels,predictedLabels)
% Using webcam to classify in real-time
camera = webcam;
preview(camera)
while(1)
    img = camera.snapshot;
    img = imresize(img,[227,227]);
    [label_new,score_live] = classify(netTransfer,img);
    imshow(img)
    title({char(label_new),num2str(max(score_live),2)});
    drawnow;

end
```

Using Webcam to classify images real time

Results: Command Window

Different classification categories. This **MATLAB** function **counts** the number of times each unique **label** occurs in the datastore.

There are 5 classification. Each Classification has 52 images count in the File path.

5×2 table

Label	Count
airplanes	52
banana	52
dollar_bill	52
faces	52
watch	52

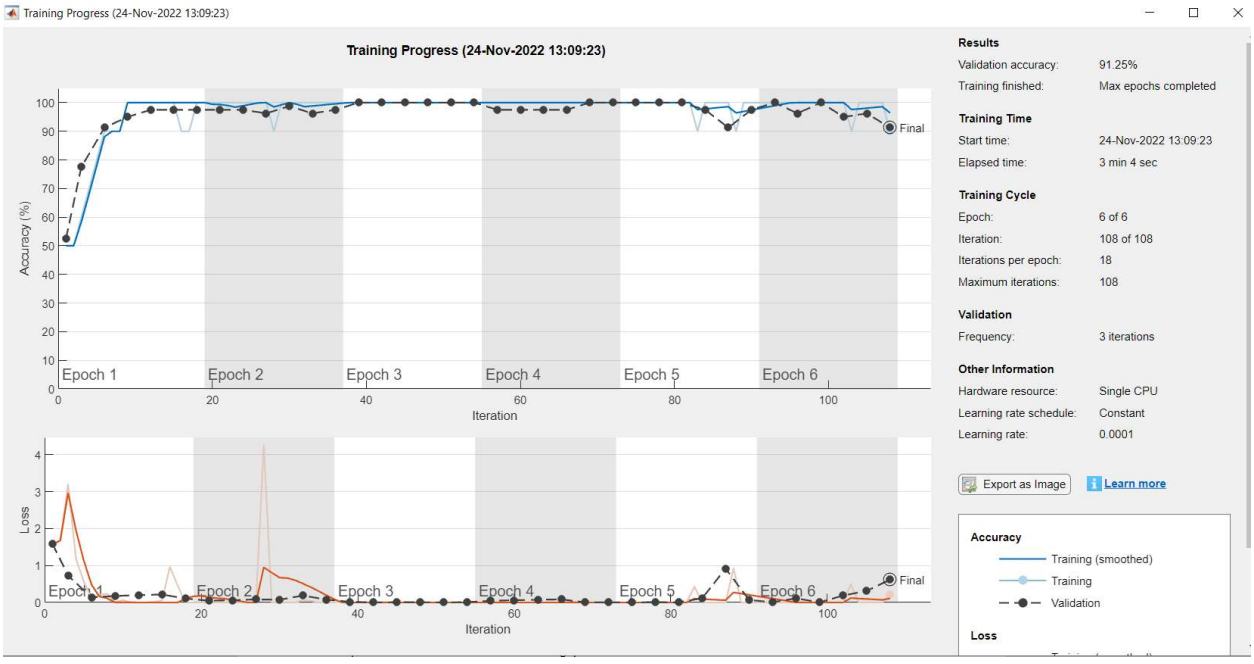
Training:

epoch - Presentation of the set of training (input and/or target) vectors to a network and the calculation of new weights and biases.

An epoch is defined as the

The below training progress shows the 6 epoch which means the number of times an algorithm visits the data set .In other words, epoch is one backward and one forward pass for all the training.

After the training is completed, We can see the validation accuracy of 91.25%



```
numClasses =

     5


netTransfer =

  SeriesNetwork with properties:

         Layers: [25×1 nnet.cnn.layer.Layer]
     InputNames: {'data'}
    OutputNames: {'classoutput'}
```

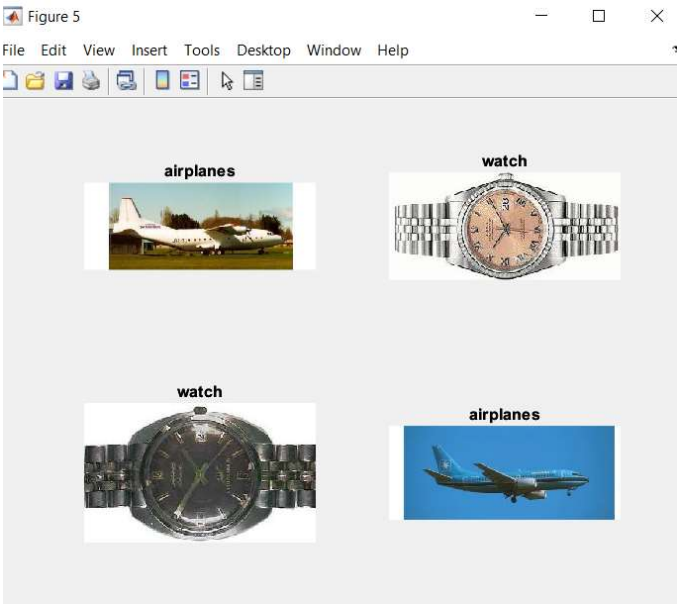
```
accuracy =

    0.9875
```

The accuracy of the dataset is 98.75%

Classification of Images:

The below images represent the results of the Neural Network.



On the **confusion matrix** plot, the rows correspond to the predicted class (Output Class) and the columns correspond to the true class (Target Class).We can see the overall accuracy is 91.3 %

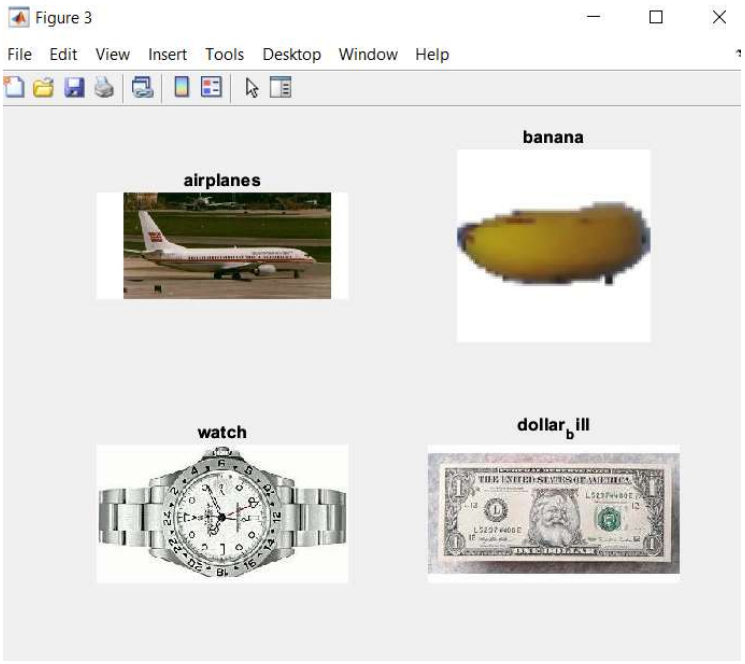
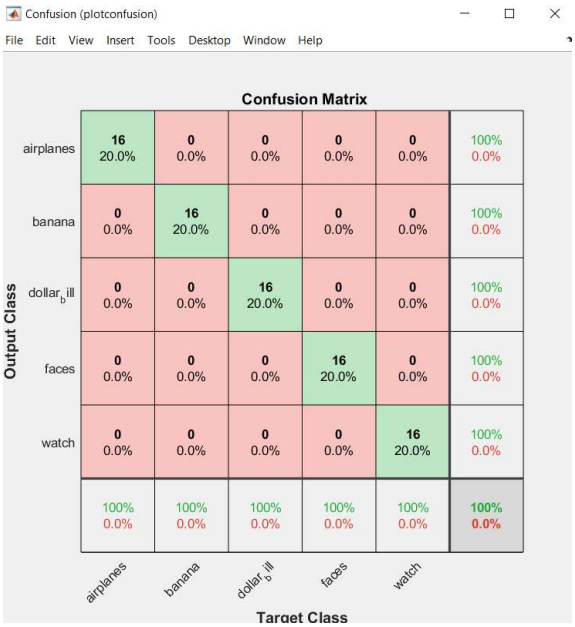


Figure 3
File Edit View Insert Tools Desktop Window Help
airplanes
banana
watch
dollar_bill


Confusion (plotconfusion)
File Edit View Insert Tools Desktop Window Help
Confusion Matrix
airplanes
banana
dollar_bill
faces
watch
Output Class
Target Class

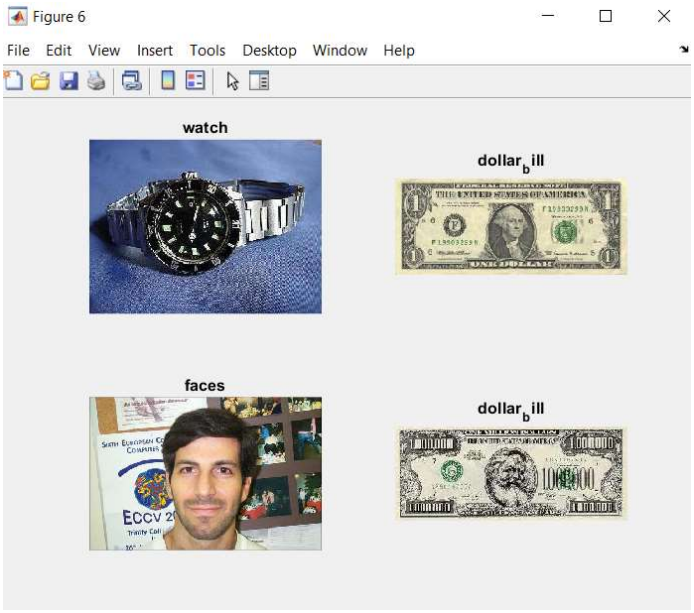
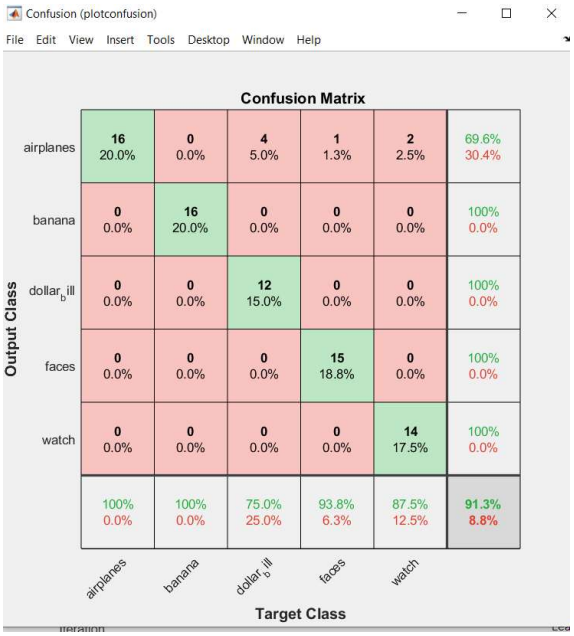


Figure 6
File Edit View Insert Tools Desktop Window Help
watch
dollar_bill
faces
dollar_bill


Confusion (plotconfusion)
File Edit View Insert Tools Desktop Window Help
Confusion Matrix
airplanes
banana
dollar_bill
faces
watch
Output Class
Target Class

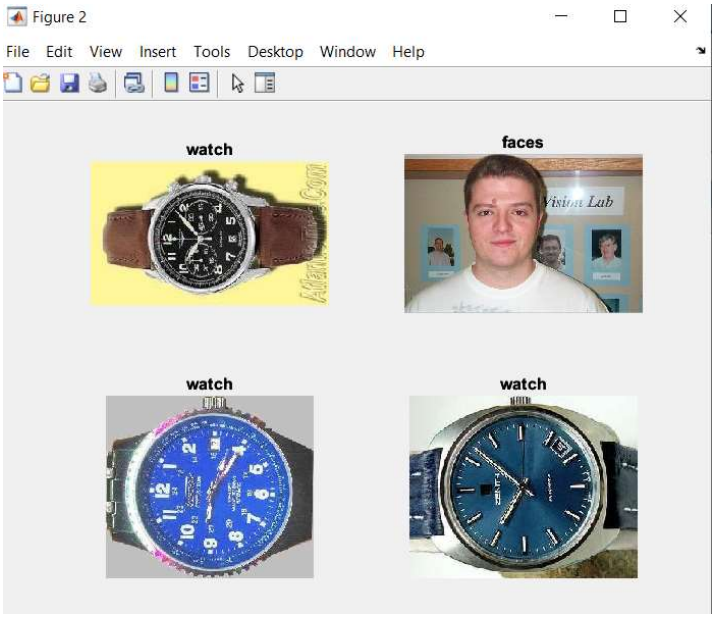


Figure 2
File Edit View Insert Tools Desktop Window Help
watch
faces
watch
watch

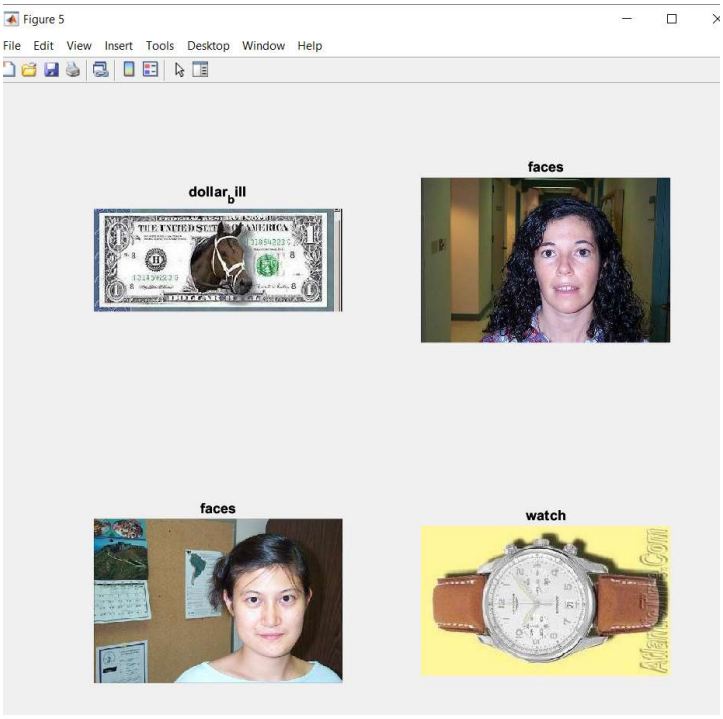
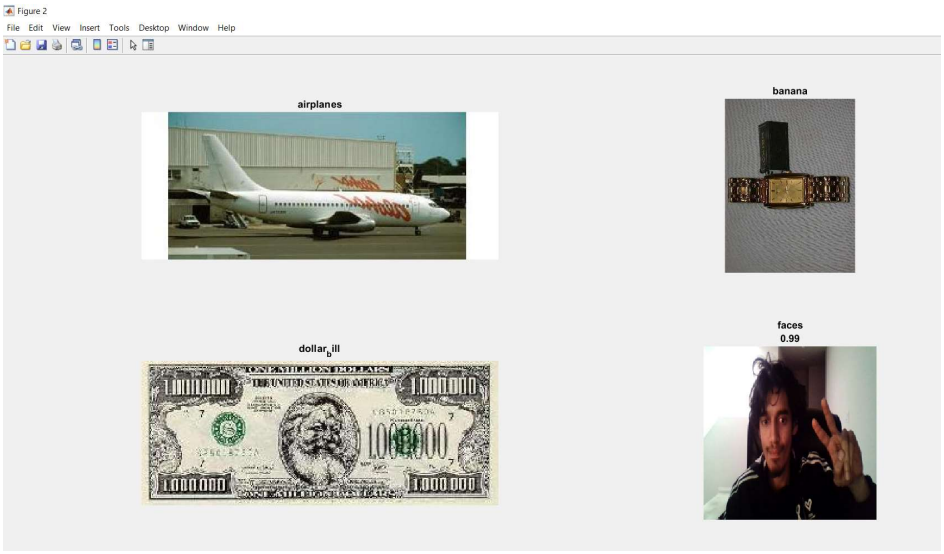
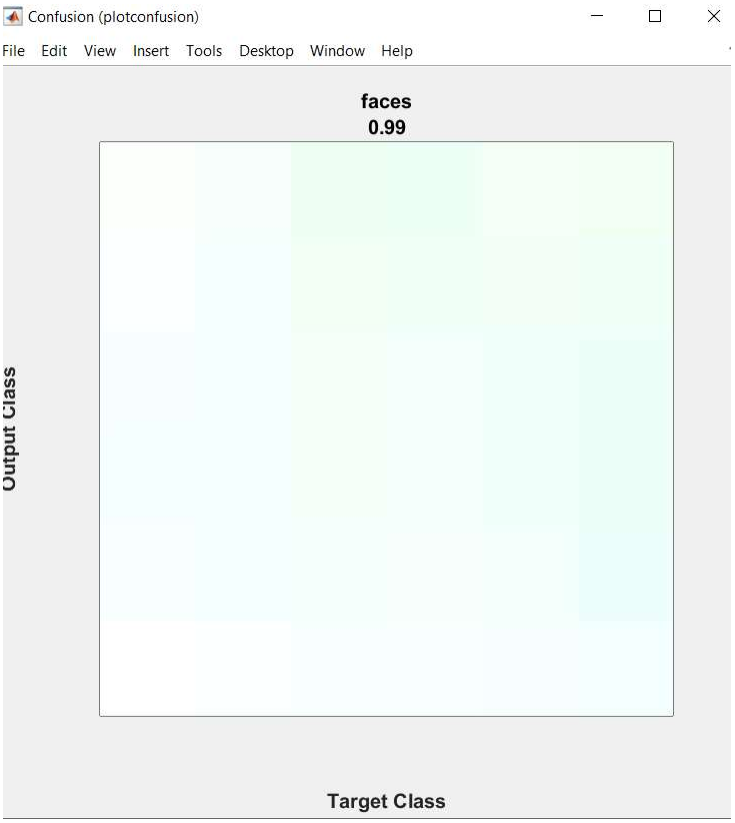


Figure 5
File Edit View Insert Tools Desktop Window Help
dollar_bill
faces
faces
watch



Dollar Bill Object recognition using live web camera



Face recognition using Live Webcamera