CECS 590 - Assignment 3 – CNN with MATLAB

Akshatha Thonnuru Swamygowda (017532175) and Sarveshwaran Sampathkumar (017387654)

Aim of the Assignment:
To train a neural network to recognize handwritten digits presented in the MNIST dataset.

Solution:
The following are the list of steps that were performed:

1. Classify an image:
Image classification is done using the following set of code:

Pre-trained AlexNet model is loaded as the first step. The code is as follows:

```
net = alexnet;
```

The image is read using imread method. The code is as follows:

```
I = imread('peppers.png');
figure
imshow(I)
```

Output is as follows:

InputSize property is used to find the input size of the network. The code and output is as follows:

```
sz = net.Layers(1).InputSize
```

Output is as follows:

```
   227   227     3
```

We then crop the image of the input size of the network. Code is as follows:

```
I = I(1:sz(1),1:sz(2),1:sz(3));
figure
imshow(I)
```

Output is as follows:

Classify method is used to classify the image. Code is as follows:

```
label = classify(net,I)
```

Output is as follows:

```
categorical
     bell pepper
```

The following code is used to show the image with its classification tag:

```
figure
imshow(I)
title(char(label))
```

Output is as follows:

2. Perform feature extraction:

The first step as part of the feature extraction is to load the data. The code for loading the data is as follows:

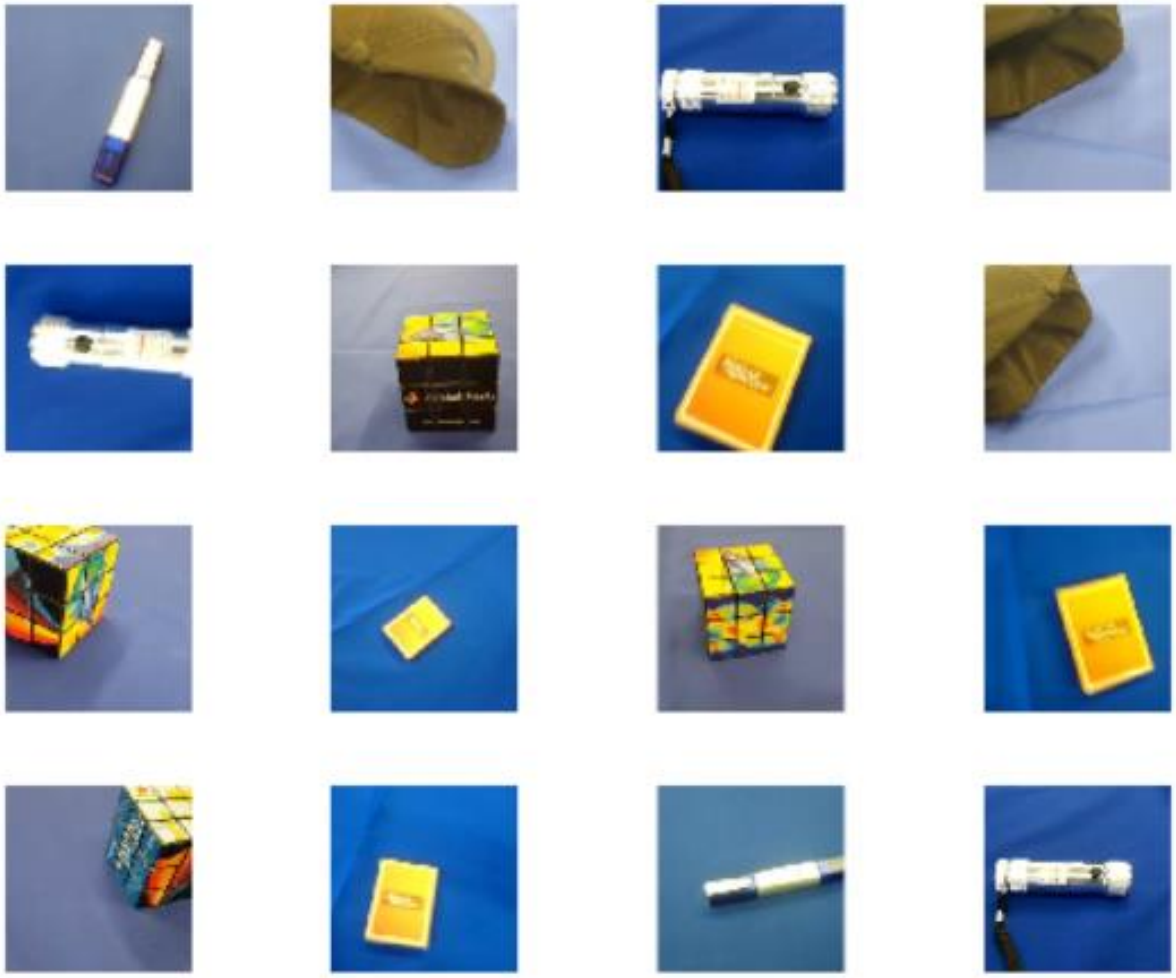
```
unzip('MerchData.zip');
imds = imageDatastore('MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');

[imdsTrain,imdsTest] = splitEachLabel(imds,0.7,'randomized');
```

To display some of the sample images, we use the following set of code:

```
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end
```

Output is as follows:



Our next step is to load the pre-trained network. The code is as follows:

```
net = alexnet;
net.Layers
```

The above code displays the network architecture. The image input layer which is the first layer requires the input images to be of size 277 by 277 by 3. Here 3 represents the number of color channels. The code is as follows:

```
inputSize = net.Layers(1).InputSize
```

The next step is to extract the image features. The code to do the image extraction is as follows:

```
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest);

layer = 'fc7';
featuresTrain = activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest = activations(net,augimdsTest,layer,'OutputAs','rows');
```

The class labels from the training and the test data are extracted using the following set of codes:

```
YTrain = imdsTrain.Labels;
YTest = imdsTest.Labels;
```

3. Use the features extracted from the training images as predictor variables

The thirst step is to use the features extracted from the training images as predictor variables. We also make use of fitceco function to fit a multiclass Support Vector Machine. The code is as follows:

```
classifier = fitcecoc(featuresTrain,YTrain);
```

Using the features extracted from the test images using the Support Vector Machine model, we then classify the images using the following code:

```
YPred = predict(classifier,featuresTest);
```

4. Classify test images

As shown in the above step, we use the predict function to classify the image. Once we have classified the images, we use the following set of code to display four sample images with their predicted labels.

```
idx = [1 5 10 15];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsTest,idx(i));
    label = YPred(idx(i));
    imshow(I)
    title(char(label))
end
```

The output is as follows:

The classification accuracy of the test set is calculated using the mean function. The mean function is applied on YPred and YTest variable. The code is follows:

```
accuracy = mean(YPred == YTest)
```

The output for above code is:

```
accuracy = 0.95
```

5. Transfer learning – transfer layers to new network

In this step, we manually captured the photos of multiple objects (example: Sweat shirt, Name Tag etc) using the phone camera and the images were transferred to the laptop. This was used as the input for this assignment. We loaded these images using the following set of code:

```
imds = imageDatastore('data', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');
```

6. Train the new network

Once we load the images, our next step is to train them. The code is as follows:

Splitting is done using the following code:

```
[imdsTrain,imdsTest] = splitEachLabel(imds,0.7,'randomized');
```

Sample images are displayed using following code:

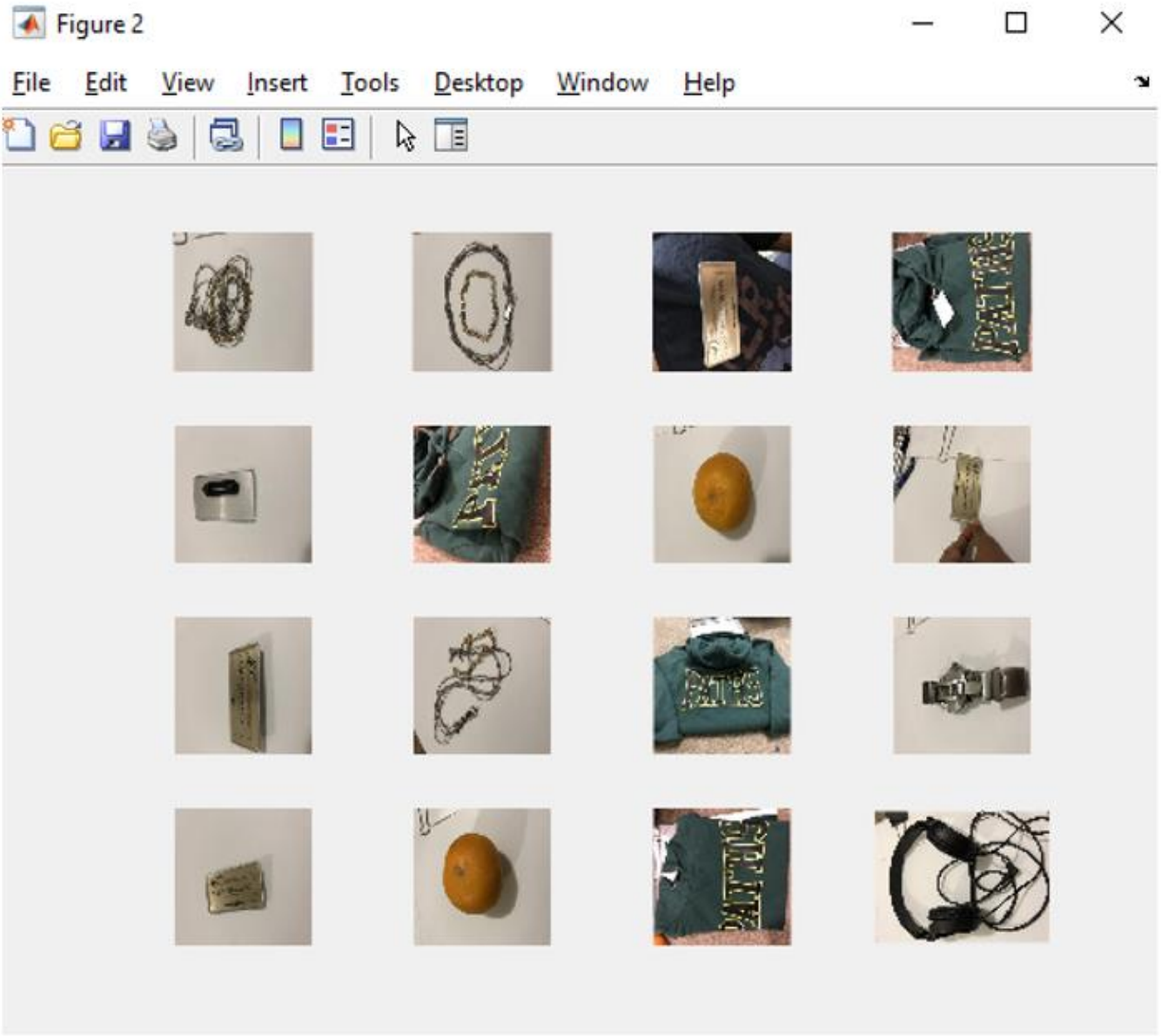
```
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
```

```
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end
```

The explanation for the below set of code is like the one as explained in step 2.

```
net = alexnet;
net.Layers;
inputSize = net.Layers(1).InputSize;
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest);
layer = 'fc7';
featuresTrain = activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest = activations(net,augimdsTest,layer,'OutputAs','rows');
YTrain = imdsTrain.Labels;
```

The following is the output which displays all the sample images which we used for this assignment.

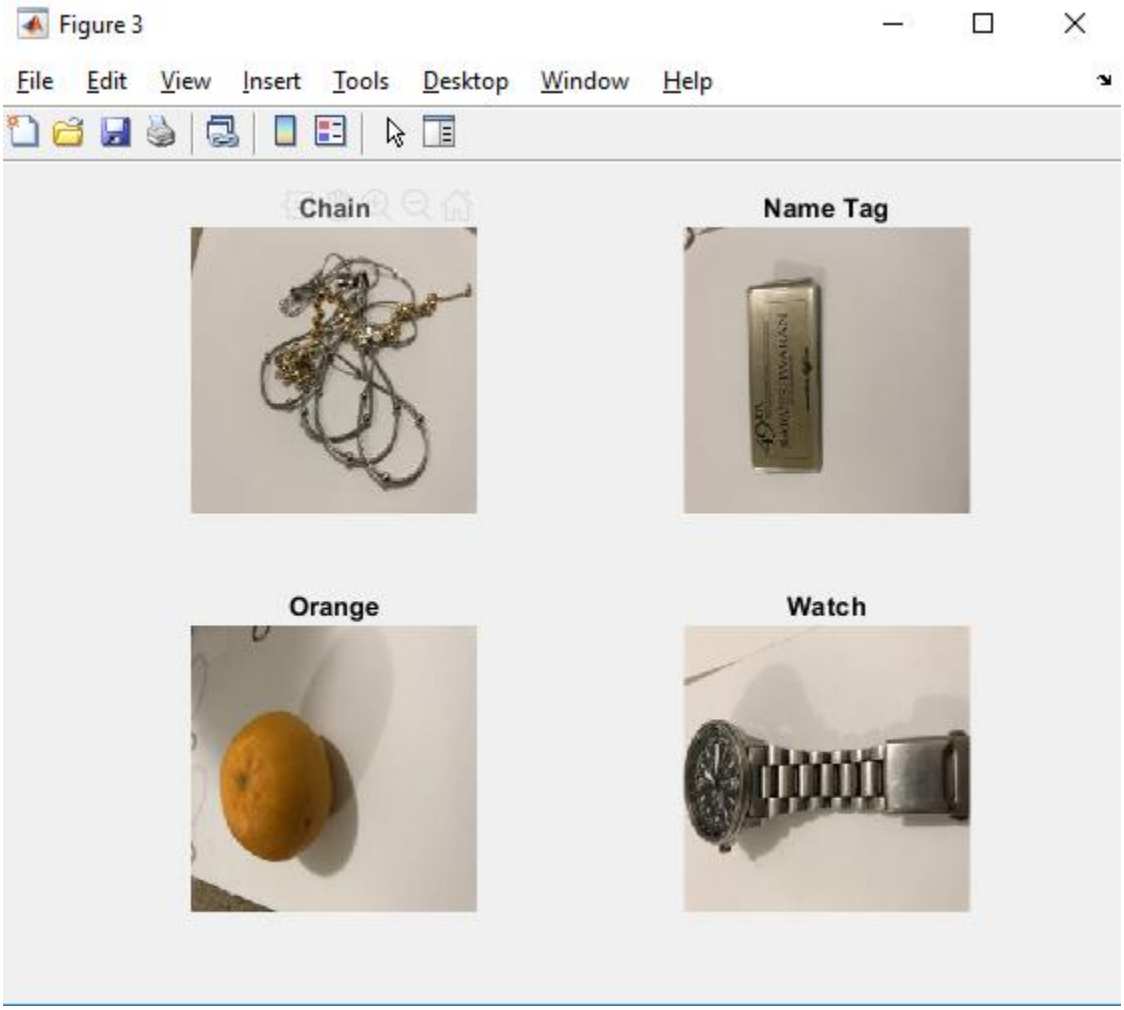


7. Create a new dataset

For this assignment, we use the same set of images which we captured using the phone and transferred to the laptop. After we trained the network in the previous step, we run the below set of code to test if the model is able to identify the images correctly.

```
YTest = imdsTest.Labels;
classifier = fitcecoc(featuresTrain,YTrain);
YPred = predict(classifier,featuresTest);
idx = [1 5 10 15];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsTest,idx(i));
    label = YPred(idx(i));
    imshow(I)
    title(char(label))
end
accuracy = mean(YPred == YTest);
```

The output is as follows:



The accuracy ranged between 0.9 and 1.0. From the above we can see that the objects are identified correctly.

8. Analytical Data:

The below shows the training profess of the model. From the accuracy graph we can see that the accuracy is increasing for every epoch. And also, we can see from the Loss graph that the Loss percentage is decreasing for every epoch. By combining both the graph we can see that as the accuracy was increasing, the loss was decreasing. This shows that the model has been trained properly.