COMP 5413 FB - Topics in Machine Learning

Final Project Report

Komal Barge (1095709)

Devanshu Mishra (1104093)

Contents:

Specifications:

Platform: MATLAB

Operating System: Windows 10 (Google Cloud Instance)

GPU: Nvidia Tesla K80

Implementation:

Image Preprocessing:

We generated the images out of the datasets (into the required dimensions of deep convolutional network) to take less time for running a program each time and avoided the data array conversion timing which is significantly higher than the image conversion timing.

1. CIFAR-10

Dataset Information:

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.[4]

Methods Used:

We have used pre-trained GoogLeNet.

GoogLeNet has been trained on over a million images and can classify images into 1000 object categories (such as keyboard, coffee mug, pencil, and many animals). The network has learned rich feature representations for a wide range of images. The network takes an image as input, and then outputs a label for the object in the image together with the probabilities for each of the object categories.[5]

Code:

```
%% Load Googlenet pretrained neural network
network = googlenet;
layers = network.Layers;
lgraph = layerGraph(network);

%% Replace last 3 layers in GoogLeNet network
%% Number of classes provided to fully connected layer is 10
lgraph = replaceLayer(lgraph,'loss3-classifier', fullyConnectedLayer(10,'Name','fcLayer','WeightLearnRateFactor',20,'BiasLearnRateFactor', 20));
lgraph = replaceLayer(lgraph,'prob',softmaxLayer('Name','smLayer'));
lgraph = replaceLayer(lgraph,'output',classificationLayer('Name','outputLayer'));

%% Load images for CIFAR10 dataset
Ximds = imageDatastore('CIFAR-10\TRAIN\','IncludeSubfolders',true,'LabelSource','foldernames');
Xtestimds = imageDatastore('CIFAR-10\TEST\','IncludeSubfolders',true,'LabelSource','foldernames');

%% Data augmentation for images to fit into googlenet
X = augmentedImageDatastore([224 224],Ximds);
Xtest = augmentedImageDatastore([224 224],Xtestimds);

%% Options to be provided for network
miniBatchSize = 120;
opts = trainingOptions('sgdm',...
    'MiniBatchSize',miniBatchSize,... %set mini batch size
     'LearnRateSchedule','piecewise',...
     'LearnRateDropFactor',0.1,...
     'LearnRateDropPeriod',3,...
     'MaxEpochs',10,...
     'InitialLearnRate',1e-3,...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','auto');
start = cputime;

%% Train network
network = trainNetwork(X, lgraph, opts);
endtime=cputime;
predictedlabels = classify(network, Xtest);

%% Accuracy for pre-trained network model
accuracy = mean(predictedlabels == Xtestimds.Labels);
fprintf('Accuracy for pretrained GoogLeNet network for CIFAR-10: %s \n', accuracy);
fprintf('Time required for training a network: %s \n', endtime-start);
```

Figure 1: CIFAR-10 Code

Results:

Accuracy for each run:

Run 1: 93.88%

Run 2: 93.84%

Run 3: 93.73%

Average Top-1 Accuracy for CIFAR-10: 93.82%

Average Training Time for CIFAR-10: ~ 2 hours

Output Screenshots:

Run 1

```
Accuracy of ImageNet Pretrained GoogLeNet: 9.388000e-01
```

Figure 2: CIFAR-10 Run 1 Accuracy

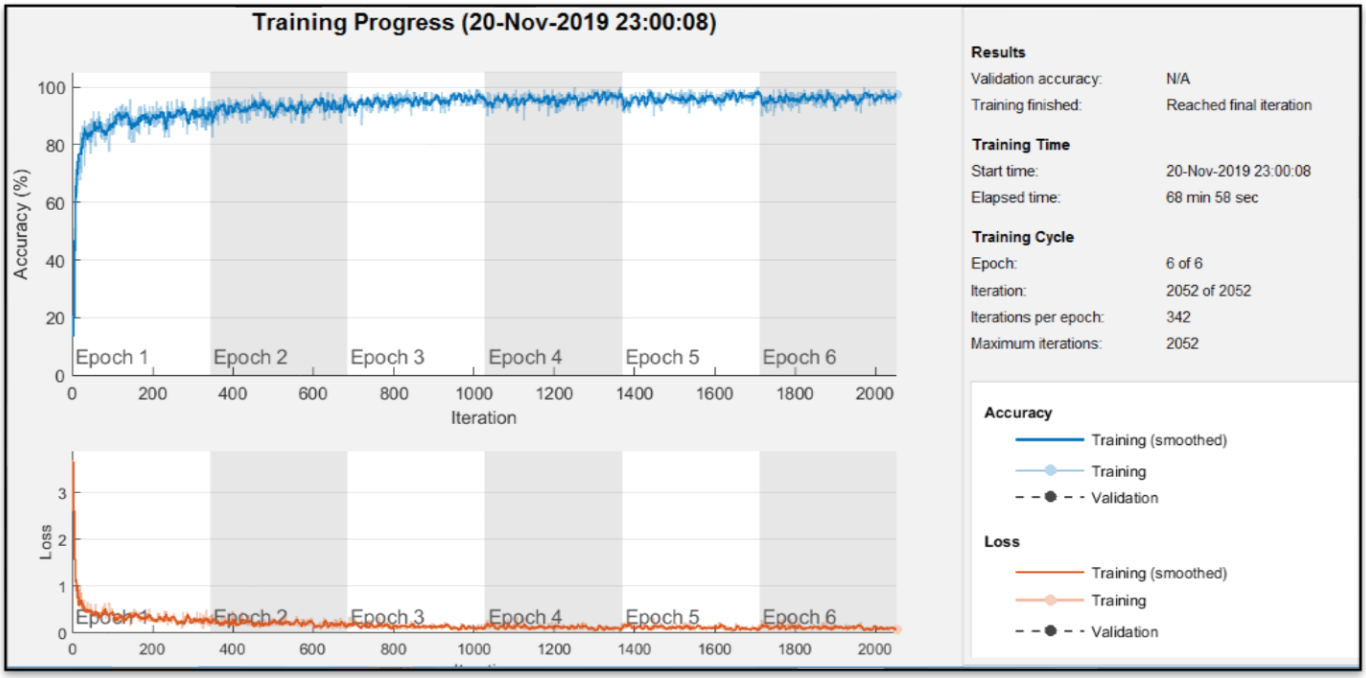

Figure 3: CIFAR-10 Run 1 Training Progress

Run 2

```
Accuracy for pretrained GoogLeNet network for CIFAR-10: 9.384000e-01
```

Figure 4: CIFAR-10 Run 2 Accuracy

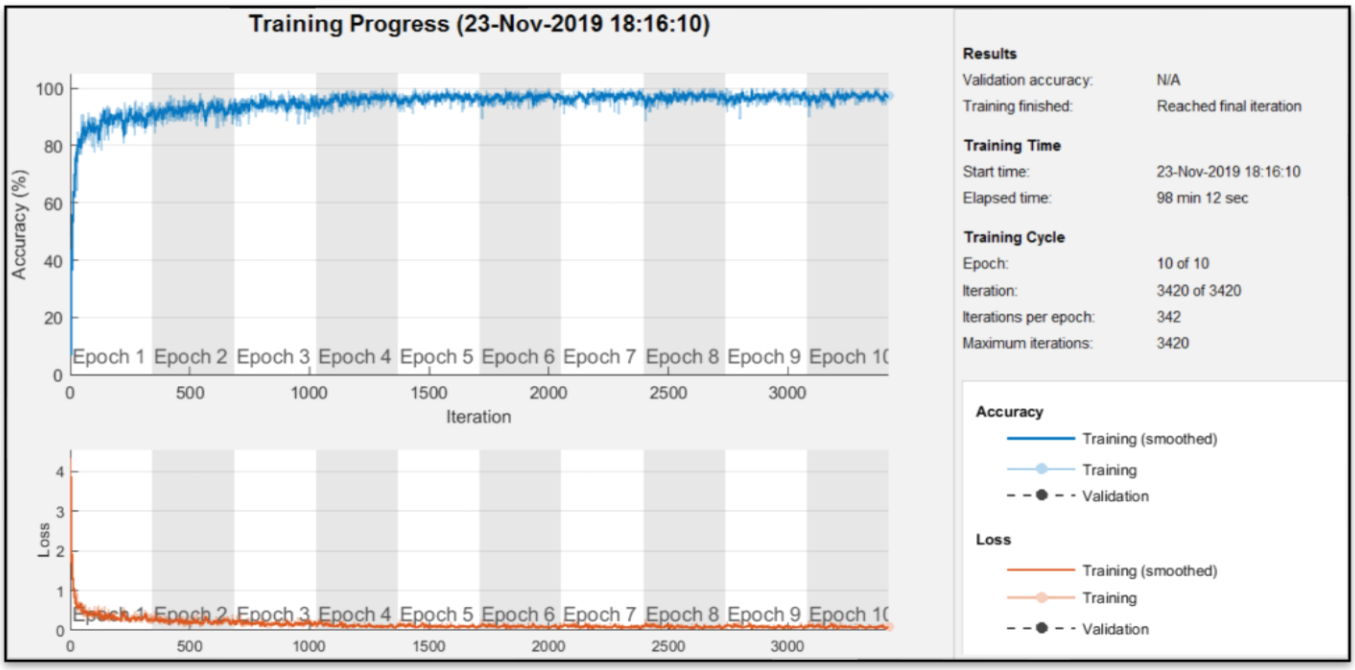

Figure 5: CIFAR-10 Run 2 Training Progress

Run 3

```
Accuracy for pretrained GoogLeNet network for CIFAR-10: 9.373000e-01
```

Figure 6: CIFAR-10 Run 3 Accuracy

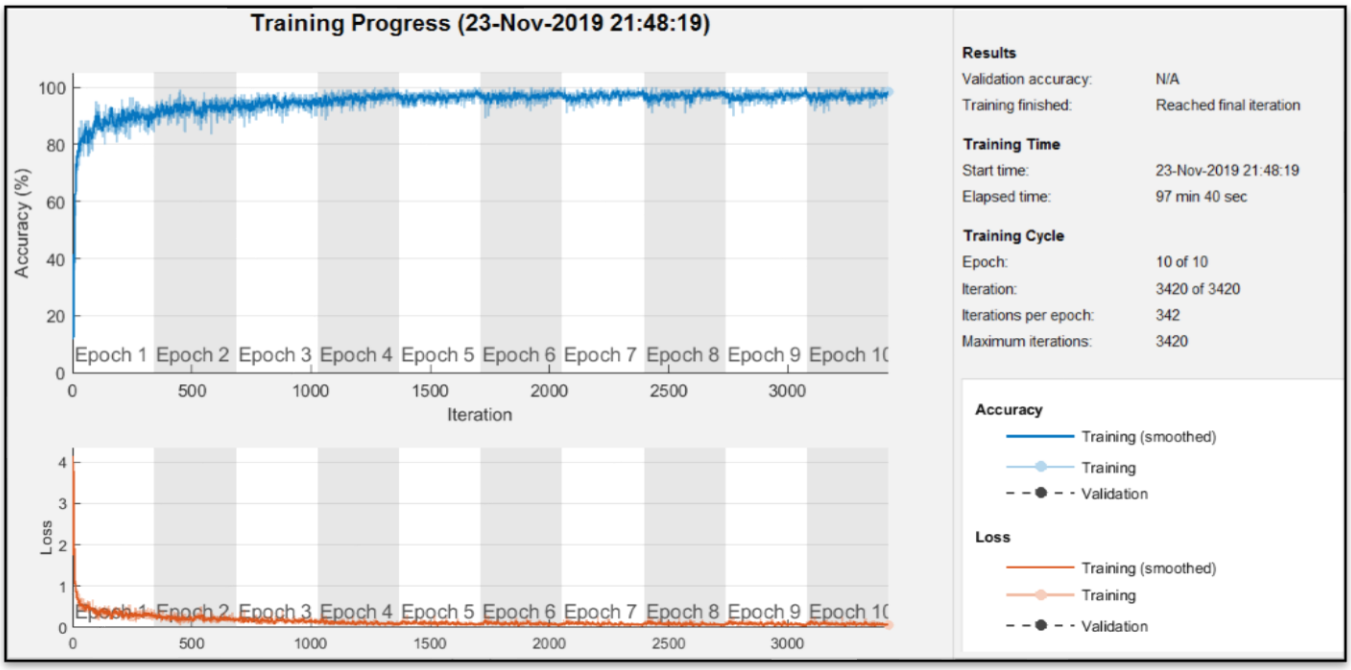

Figure 7: CIFAR-10 Run 3 Training Progress

2. CALTECH 101

Dataset Information:

Pictures of objects belonging to 101 categories. About 40 to 800 images per category. Most categories have about 50 images. Collected in September 2003 by Fei-Fei Li, Marco Andreetto, and Marc 'Aurelio Ranzato. The size of each image is roughly 300 x 200 pixels.

Method Used:

We have used combination of ResNet101's high level features and Xception convolutional network's high-level features from last fully connected layer and average pooling layer respectively. And we used Extreme learning machine (ELM) for classifying the Caltech datasets.

ResNet-101 is a pre-trained model that has been trained on a subset of the ImageNet database. The model is trained on more than a million images, has 347 layers in total, corresponding to a 101-layer residual network, and can classify images into 1000 object categories (e.g. keyboard, mouse, pencil, and many animals). [1]

Xception is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 71 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299. [2]

Extreme Learning Machine are also known as feedforward neural networks which are used for classification, regression, clustering, sparse approximation, compression and feature learning with a single layer or multiple layers of hidden nodes, where the parameters of hidden nodes (not just the weights connecting inputs to hidden nodes) need not be tuned. These hidden nodes can be randomly assigned and never updated (i.e. they are random projection but with nonlinear transforms), or can be inherited from their ancestors without being changed. In most cases, the output weights of hidden nodes are usually learned in a single step, which essentially amounts to learning a linear model. [3]

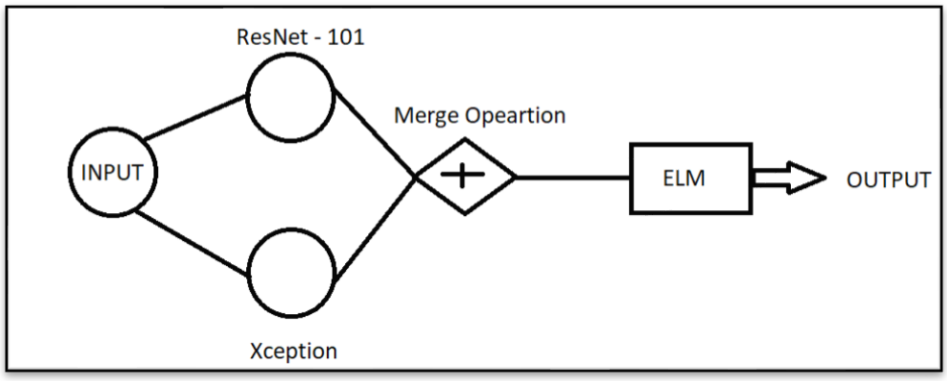

Figure 8: Network used for Caltech 101

Code:

```
%% Fetch the images from Caltech101 dataset
imageData = imageDatastore('Caltech101\101_ObjectCategories',...
'LabelSource', 'foldernames', 'IncludeSubfolders', true);

%% GPU device used for this program
disp('GPU device used for this program');
gpuDevice(1)

%% Split the labels
% Use 30 images for training and remaining images for testing purpose
disp('1. Spliting labels for training and testing');
[trainingSet, testingSet] = splitEachLabel(imageData, 30);

%% Load Resnet101
disp('2. Loading Pretrained Resnet101');
net = resnet101;

%% Redefining read function to process images
disp('3. Preprocessing images for Resnet101 network');
trainingSet.ReadFcn = @(filename)PrepImageWithResnet101Dim(filename);
testingSet.ReadFcn = @(filename)PrepImageWithResnet101Dim(filename);

 %% Get features from Resnet101
disp('4. Extracting Resnet101 deeper layer features');
extractionLayer = 'fc1000';
resnet101_trainFeatures = activations(net,trainingSet,extractionLayer,'MiniBatchSize',120);
resnet101_trainFeatures = reshape(resnet101_trainFeatures,[1*1*1000,size(resnet101_trainFeatures,4)])' ;
resnet101_testFeatures = activations(net,testingSet,extractionLayer,'MiniBatchSize',120);
resnet101_testFeatures = reshape(resnet101_testFeatures,[1*1*1000,size(resnet101_testFeatures,4)])';

%% Load xception neural network
disp('5. Loading xception neural network');
net = xception;

%% Redefining read function to process images
disp('6. Preprocessing images for xception network');
trainingSet.ReadFcn = @(filename)PrepImageWithXceptionDim(filename);
testingSet.ReadFcn = @(filename)PrepImageWithXceptionDim(filename);

%% Get training set deep features from xception
disp('7. Extracting xception features');
```

Figure 10: Clatech 101 code

```
xception_testFeatures = activations(net,testingSet,'avg_pool','MiniBatchSize',120);
xception_trainFeatures = reshape(xception_trainFeatures,[1*1*2048,size(xception_trainFeatures,4)])' ;
xception_testFeatures = reshape(xception_testFeatures,[1*1*2048,size(xception_testFeatures,4)])';

%% Merge Resnet and xception deep features for training and testing
disp('8. Merging the features from Resnet101 and xception network');
Mergedtrain = horzcat(xception_trainFeatures, resnet101_trainFeatures);
Mergedtest = horzcat(xception_testFeatures, resnet101_testFeatures);
train_labels = grp2idx(trainingSet.Labels);
test_labels = grp2idx(testingSet.Labels);

%% Creating training and testing dataset
training = horzcat(train_labels,Mergedtrain);
testing = horzcat(test_labels,Mergedtest);

disp('9. Classification using ELM algorithm');
[TrainingTime, TestingAccuracy,Training,Testing] = ELM(training, testing, 1, 10000, 'sig');
```

Figure 9: Clatech 101 code

Results:

Accuracy for each runs:

Run 1: 90.09%

Run 2: 90.29%

Run 3: 89.99%

Average Top-1 Accuracy for Caltech101: 90.12%

Average Training Time for Caltech101: ~ 0.5 hour

Output Screenshots:

Run 1

```
TrainingTime =

   32.1406


TestingTime =

    9.5313


TrainingAccuracy =

     1


TestingAccuracy =

    0.9009
```

Run 2

```
TrainingTime =

   31.3438


TestingTime =

    9.5000


TrainingAccuracy =

     1


TestingAccuracy =

    0.9029
```

Run 3

```
TrainingTime =

   32.4375


TestingTime =

    9.1406


TrainingAccuracy =

     1


TestingAccuracy =

    0.8999
```

Figure 11: Caltech 101 Accuracy and Training Time

3. CALTECH 256

Dataset Information:

There are 30,607 images in this dataset spanning 257 object categories. Object categories are extremely diverse, ranging from grasshopper to tuning fork. The distribution of images per category are:

- Min: 80
- Med: 100
- Mean: 119
- Max: 827

Methods Used:

We have used same combination as of Caltech 101 for Caltech256, ResNet101 and Xception as convolution network's deep layer features and Extreme learning machine was used for classifying the dataset.

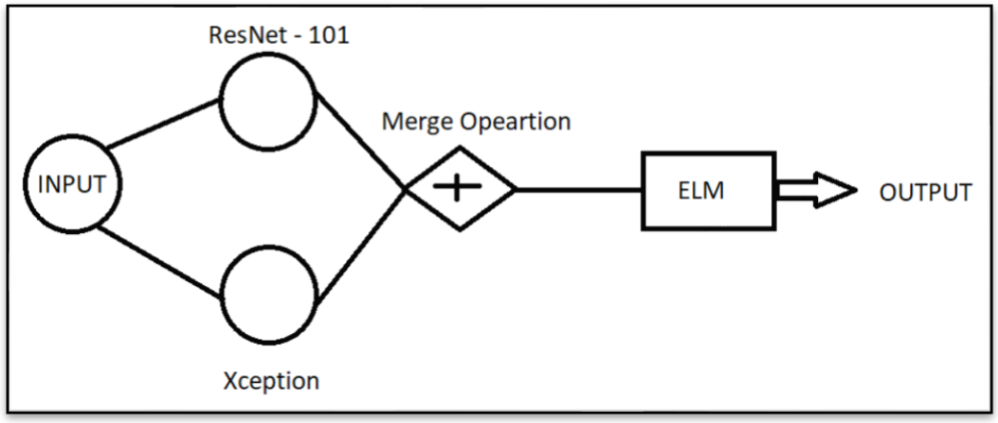

Figure 12: Network model Caltech 256

Code:

```
%% Fetch the images from Caltech256 dataset
imageData = imageDatastore('Caltech256\256_ObjectCategories',...
'LabelSource', 'foldernames', 'IncludeSubfolders', true);

%% GPU device used for this program
disp('GPU device used for this program');
gpuDevice(1)

%% Split the labels
% Use 30 images for training and remaining images for testing purpose
disp('1. Spliting labels for training and testing');
[trainingSet, testingSet] = splitEachLabel(imageData, 30);

%% Load Resnet101
disp('2. Loading Pretrained Resnet101');
net = resnet101;

%% Redefining read function to process images
disp('3. Preprocessing images for Resnet101 network');
trainingSet.ReadFcn = @(filename)PrepImageWithResnet101Dim(filename);
testingSet.ReadFcn = @(filename)PrepImageWithResnet101Dim(filename);

 %% Get features from Resnet101
disp('4. Extracting Resnet101 deeper layer features');
extractionLayer = 'fc1000';
resnet101_trainFeatures = activations(net,trainingSet,extractionLayer,'MiniBatchSize',120);
resnet101_trainFeatures = reshape(resnet101_trainFeatures,[1*1*1000,size(resnet101_trainFeatures,4)])' ;
resnet101_testFeatures = activations(net,testingSet,extractionLayer,'MiniBatchSize',120);
resnet101_testFeatures = reshape(resnet101_testFeatures,[1*1*1000,size(resnet101_testFeatures,4)])';

%% Load xception neural network
disp('5. Loading xception neural network');
net = xception;

%% Redefining read function to process images
disp('6. Preprocessing images for xception network');
trainingSet.ReadFcn = @(filename)PrepImageWithXceptionDim(filename);
testingSet.ReadFcn = @(filename)PrepImageWithXceptionDim(filename);

%% Get training set deep features from xception
disp('7. Extracting xception features');
```

Figure 13: Caltech 256 Code

```
xception_trainFeatures = activations(net,trainingSet,'avg_pool','MiniBatchSize',120);
xception_testFeatures = activations(net,testingSet,'avg_pool','MiniBatchSize',120);
xception_trainFeatures = reshape(xception_trainFeatures,[1*1*2048,size(xception_trainFeatures,4)])' ;
xception_testFeatures = reshape(xception_testFeatures,[1*1*2048,size(xception_testFeatures,4)])';

%% Merge Resnet and xception deep features for training and testing
disp('8. Merging the features from Resnet101 and xception network');
Mergedtrain = horzcat(xception_trainFeatures, resnet101_trainFeatures);
Mergedtest = horzcat(xception_testFeatures, resnet101_testFeatures);
train_labels = grp2idx(trainingSet.Labels);
test_labels = grp2idx(testingSet.Labels);

%% Creating training and testing dataset
training = horzcat(train_labels,Mergedtrain);
testing = horzcat(test_labels,Mergedtest);

disp('9. Classification using ELM algorithm');
[TrainingTime, TestingAccuracy,Training,Testing] = ELM(training, testing, 1, 10000, 'sig');
```

Figure 14: Caltech 256 Code

Results:

Accuracy for each run:

Run 1: 74.78%

Run 2: 74.24%

Run 3: 74.53%

Average Top-1 Accuracy for Caltech256: 74.95%

Average Training Time for Caltech256: ~ 1.5 hours

Output Screenshots:

Run 1

```
TrainingTime =

  350.4219


TestingTime =

   40.3750


TrainingAccuracy =

     1


TestingAccuracy =

    0.7478
```

Run 2

```
TrainingTime =

  337.4375


TestingTime =

   39.7656


TrainingAccuracy =

     1


TestingAccuracy =

    0.7424
```

Run 3

```
TrainingTime =

  357.0156


TestingTime =

   40.2344


TrainingAccuracy =

     1


TestingAccuracy =

    0.7453
```

Figure 15: Caltech 256 Accuracies and Training Time

4. CIFAR-100

Dataset Information:

This dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 super classes. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).[4]

Methods Used:

We have used same pre-trained network method which we have used for CIFAR-10 which is **GoogLeNet** deep convolutional network.

Code:

```
%% Load Googlenet pretrained neural network
network = googlenet;
layers = network.Layers;
lgraph = layerGraph(network);

%% Replace last 3 layers in GoogLeNet network
%% Number of classes provided to fully connected layer is 100
lgraph = replaceLayer(lgraph,'loss3-classifier', fullyConnectedLayer(100,'Name','fcLayer','WeightLearnRateFactor',20,'BiasLearnRateFactor', 20));
lgraph = replaceLayer(lgraph,'prob',softmaxLayer('Name','smLayer'));
lgraph = replaceLayer(lgraph,'output',classificationLayer('Name','outputLayer'));

%% Load images for CIFAR100 dataset
Ximds = imageDatastore('CIFAR-100\TRAIN\','IncludeSubfolders',true,'LabelSource','foldernames');
Xtestimds = imageDatastore('CIFAR-100\TEST\','IncludeSubfolders',true,'LabelSource','foldernames');

%% Data augmentation for images to fit into googlenet
X = augmentedImageDatastore([224 224],Ximds);
Xtest = augmentedImageDatastore([224 224],Xtestimds);

%% Options to be provided for network
miniBatchSize = 30;
opts = trainingOptions('sgdm',...
    'MiniBatchSize',miniBatchSize,... %set mini batch size
      'LearnRateSchedule','piecewise',...
      'LearnRateDropFactor',0.1,...
      'LearnRateDropPeriod',3,...
      'MaxEpochs',10,...
      'InitialLearnRate',1e-3,...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','auto');
start = cputime;

%% Train network
network = trainNetwork(X, lgraph, opts);
endtime=cputime;
predictedlabels = classify(network, Xtest);

%% Accuracy for pre-trained network model
accuracy = mean(predictedlabels == Xtestimds.Labels);
fprintf('Accuracy for pretrained GoogLeNet network for CIFAR-100: %s \n', accuracy);
fprintf('Time required for training a network: %s \n', endtime-start);
```

Figure 16: CIFAR-100 Code

Results:

Accuracy for each runs:

Run 1: 79.72%

Run 2: 79.44%

Run 3: 79.14%

Average Top-1 Accuracy for CIFAR-100: 79.44%

Average Training Time for CIFAR-100: ~ 3.5 hours

Output Screenshots:

Run1

```
Accuracy for pretrained GoogLeNet network for CIFAR-100: 7.972000e-01
Time required for training a network: 1.015659e+04
```

Figure 17: CIFAR-100 Run 1 Accuracy

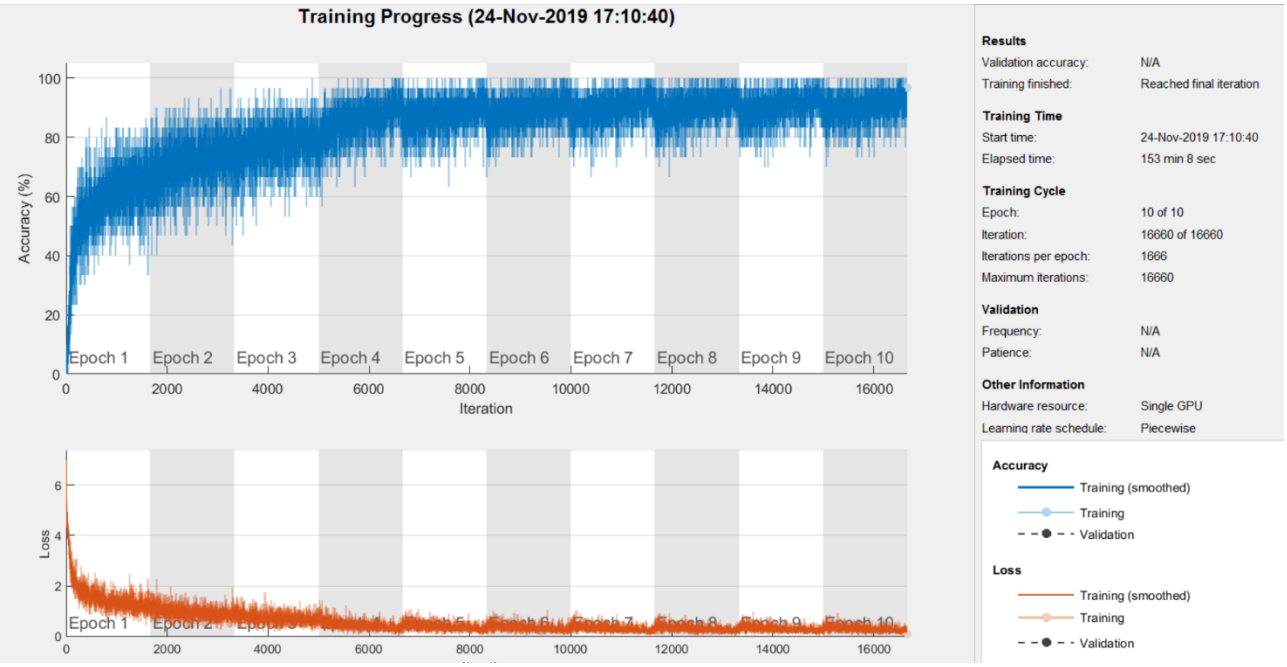

Figure 18: CIFAR-100 Run 1 Training Progress

Run 2

```
Accuracy for pretrained GoogLeNet network for CIFAR-100: 7.944000e-01
Time required for training a network: 1.121927e+04
```

Figure 19: CIFAR-100 Run 2 Accuracy

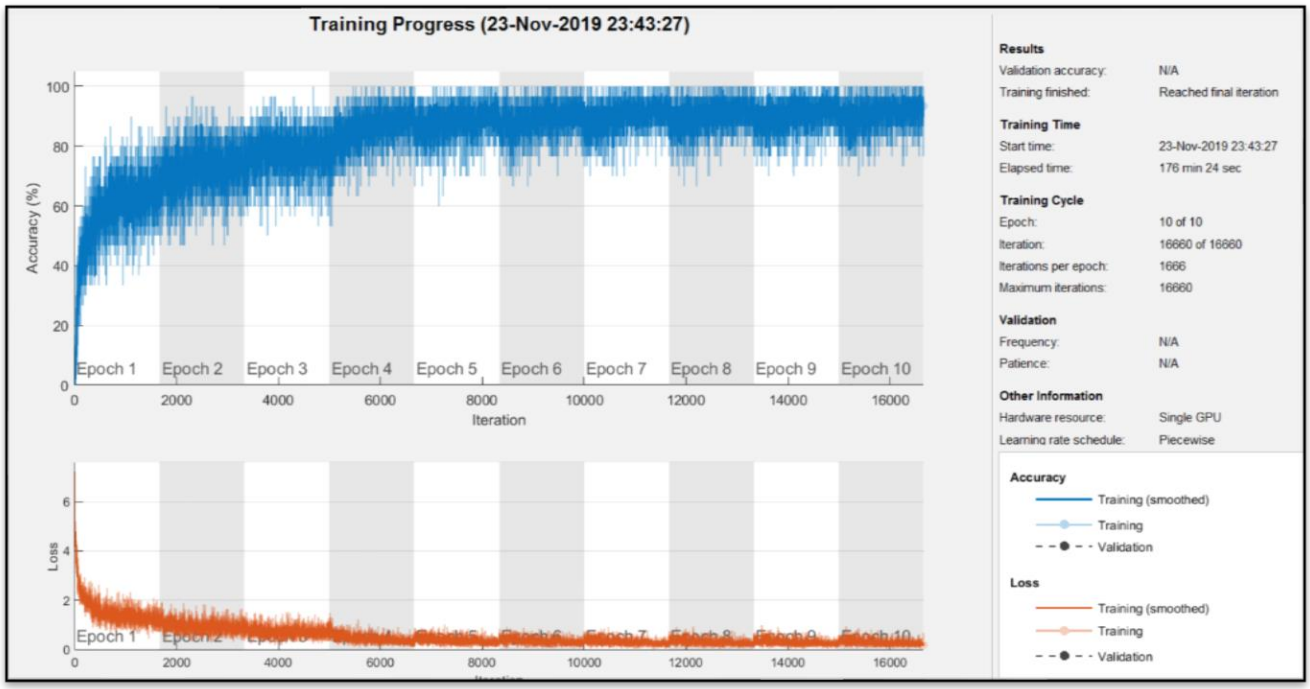

Figure 20: CIFAR-100 Run 2 Training Progress

Run 3

```
Accuracy for pretrained GoogLeNet network for CIFAR-100: 7.914000e-01
Time required for training a network: 1.173291e+04
```

Figure 21: CIFAR-100 Run 3 Accuracy

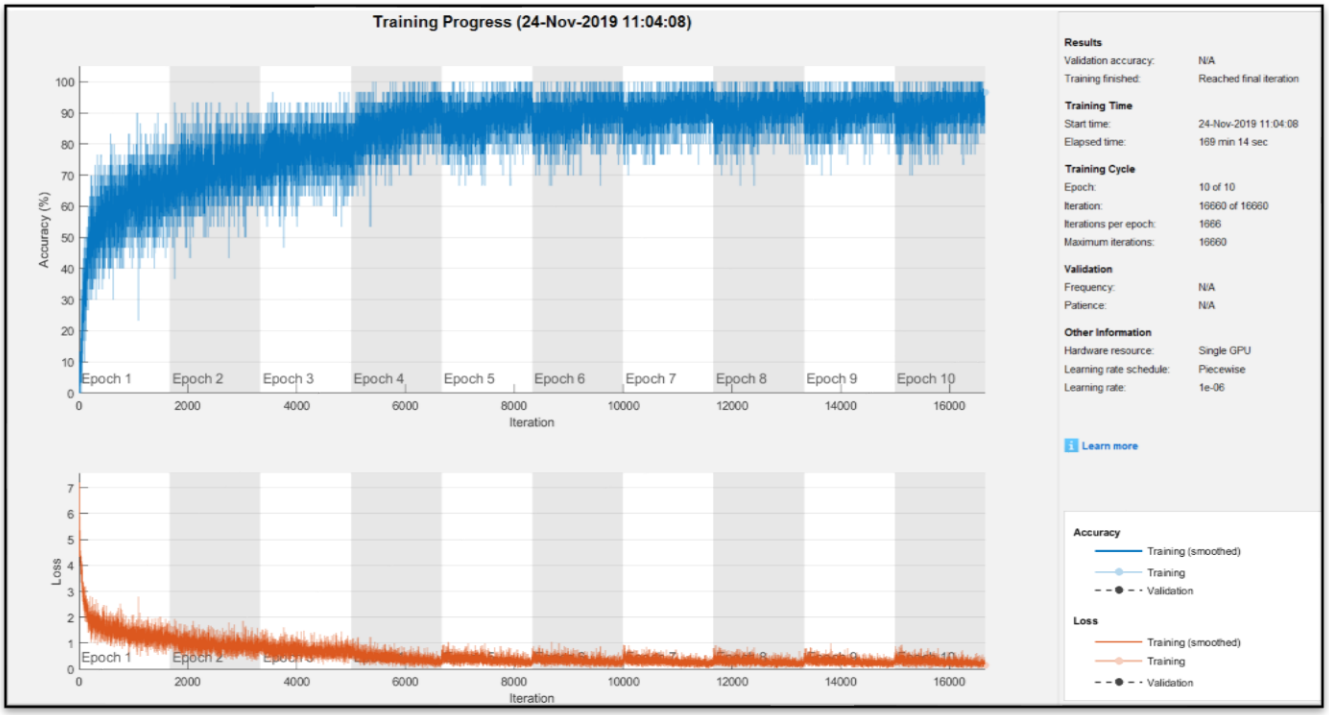

Figure 22: CIFAR-100 Run 3 Training Accuracy

Average Top-1 accuracy for Object-centric Image Recognition task :

(93.88 + 90.29 + 74.95 + 79.72)/4 = 84.75%

References:

[1] https://www.mathworks.com/matlabcentral/fileexchange/65678-deep-learning-toolbox-model-for-resnet-101-network

[2] https://www.mathworks.com/help/deeplearning/ref/xception.html

[3] https://en.wikipedia.org/wiki/Extreme_learning_machine

[4] https://www.cs.toronto.edu/~kriz/cifar.html

[5] https://www.mathworks.com/help/deeplearning/ref/googlenet.html