Masters in Computer Science

Topics in Deep Learning

(COMP-5413)

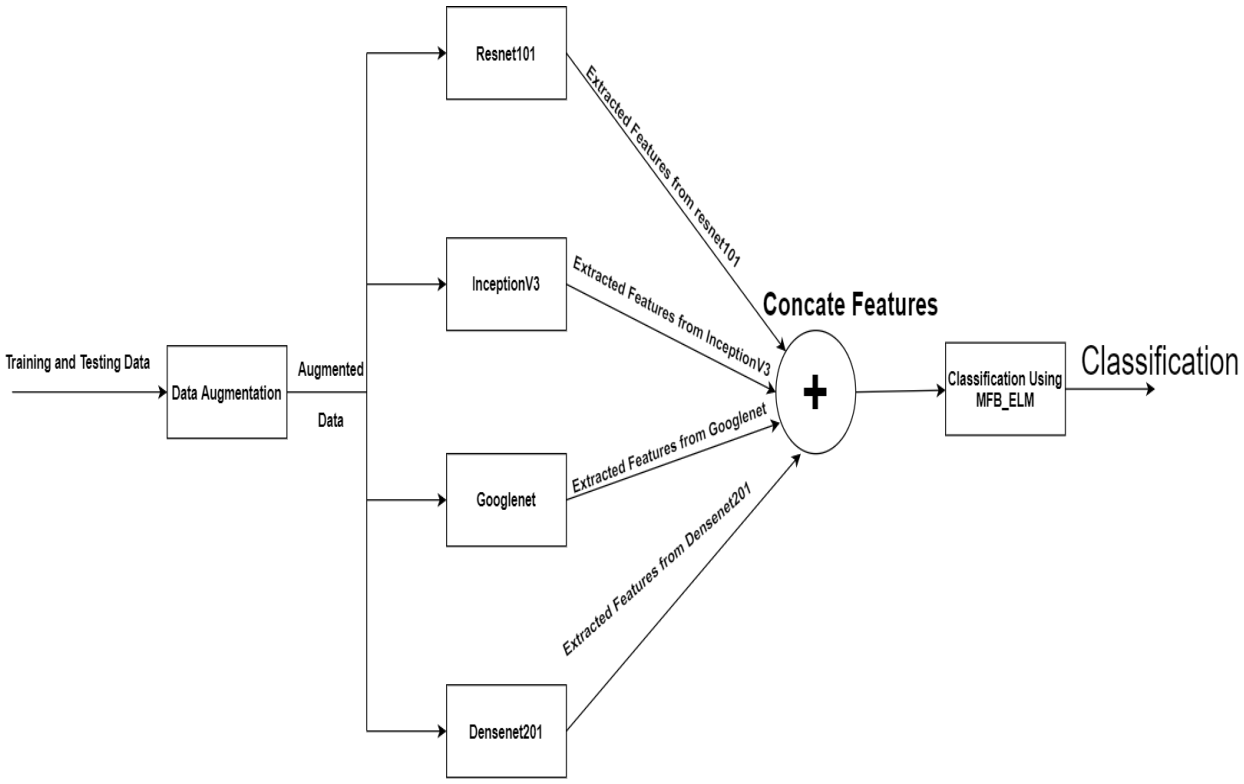
Group:

1. Jaykumar Nariya (1116571)
2. Neel Zadafiya (1115533)

Project:

Task1: Object-centric Image recognition using Caltech101, Caltech256, Cifar10.

Methodology:



The Following structure or Network is used for classification for all three datasets. We obtained deep features from Resnet101, InceptionV3, Googlenet and Densenet201. These Deep Features are then Combined. We used MFB ELM is used as classifier and trained on using data and used for classification. The Output of the class is provided by the MFB_ELM.

1. Caltech101

Code:

```
tic
% define  folder
Folder = fullfile('C:\Users\Neel\Desktop\DLProject\data', 'caltech101');
disp('steps to output')

% Create ImageDatastore of the dataset for processing in Matlab.
rootFolder = fullfile(Folder, '101_ObjectCategories');
imds = imageDatastore(fullfile(rootFolder), 'LabelSource',
'foldernames','IncludeSubfolders',true);
clear Folder rootFolder;

% Split each label Using 30 images for training and rest for testing
[trainingSet, testingSet] = splitEachLabel(imds, 30);

% Load resnet
disp('1. Loading Pretrained Resnet');
 net = resnet101;

 %set imageSize according to DCNN first input layer
 imageSize = net.Layers(1).InputSize;

 disp('2.Data Augmentatiojn');
%Define Augmentation function arguments
imageAugmenter = imageDataAugmenter('RandXReflection',true,...
    'RandYReflection',false,...
    'RandRotation',[0 0],...
    'RandScale',[1 1],...
    'RandXTranslation',[0 0],...
    'RandYTranslation',[0 0]);

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);
%resetgpu
gpuDevice(1)
```

```
% Get features from resnet
 disp('3. Loading Resnet train features');
 %Extract training features from resnet101 DCNN
 resnet_features_train =
activations(net,augImdstraining,'fc1000','MiniBatchSize',200);

 disp('4. Loading Resnet test features');
 %Extract testing features from resnet101 DCNN
 resnet_features_test =
activations(net,augImdstesting,'fc1000','MiniBatchSize',200);

 %Reshape training and testing features from resnet101
 resnet_features_train =
reshape(resnet_features_train,[1*1*1000,size(resnet_features_train,4)])' ;
 resnet_features_test =
reshape(resnet_features_test,[1*1*1000,size(resnet_features_test,4)])';

 disp('5. Loading Pretrained inceptionv3');
% Load inceptionv3
net = inceptionv3;

%set imageSize according to DCNN first input layer
imageSize = net.Layers(1).InputSize;

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb' ,'DataAugmentation',imageAugmenter);

%resetgpu
gpuDevice(1)

disp('6. Loading inceptionv3 train features');
 %Extract training features from inceptionv3 DCNN
inceptionv3_features_train =
activations(net,augImdstraining,'avg_pool','MiniBatchSize',200);

disp('7. Loading inceptionv3 test features');
 %Extract testing features from inceptionv3 DCNN
inceptionv3_features_test =
activations(net,augImdstesting,'avg_pool','MiniBatchSize',200);

%Reshape training and testing features from inceptionv3
inceptionv3_features_train =
reshape(inceptionv3_features_train,[1*1*2048,size(inceptionv3_features_train,
4)])' ;
inceptionv3_features_test =
reshape(inceptionv3_features_test,[1*1*2048,size(inceptionv3_features_test,4)
])';

disp('8. Loading Pretrained googlenet');
% Load inceptionv3
```

```
% Load inceptionv3
net = googlenet;

%set imageSize according to DCNN first input layer
imageSize = net.Layers(1).InputSize;

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);
%resetgpu
gpuDevice(1)

 disp('9. Loading googlenet train features');
 %Extract training features from googlenet DCNN
  googlenet_features_train = activations(net,augImdstraining,'loss3-
classifier','MiniBatchSize',200);

  disp('10. Loading googlenet test features');
  %Extract testing features from googlenet DCNN
googlenet_features_test = activations(net,augImdstesting,'loss3-
classifier','MiniBatchSize',200);

%Reshape training and testing features from googlenet
googlenet_features_train =
reshape(googlenet_features_train,[1*1*1000,size(googlenet_features_train,4)])
' ;
googlenet_features_test =
reshape(googlenet_features_test,[1*1*1000,size(googlenet_features_test,4)])';

disp('11. Loading Pretrained densenet201');
% Load densenet201
net = densenet201;

%set imageSize according to DCNN first input layer
imageSize = net.Layers(1).InputSize;

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%resetgpu
gpuDevice(1)

 disp('12. Loading densenet train features');
 %Extract training features from densenet DCNN
```

```
densenet_features_train =
activations(net,augImdstraining,'fc1000','MiniBatchSize',200);

  disp('13. Loading densenet test features');
  %Extract testing features from densenet DCNN
densenet_features_test =
activations(net,augImdstesting,'fc1000','MiniBatchSize',200);

%Reshape training and testing features from densenet
densenet_features_train =
reshape(densenet_features_train,[1*1*1000,size(densenet_features_train,4)])'
;
densenet_features_test =
reshape(densenet_features_test,[1*1*1000,size(densenet_features_test,4)])';


disp('14. Combining the training features from All DCNN');
% Merge Resnet and googlenet deep features for training
x = horzcat(resnet_features_train,googlenet_features_train);
% Merge densenet and inceptionv3 deep features for training
w = horzcat(inceptionv3_features_train, densenet_features_train);
% Merge all deep features for training
new_F_train = horzcat(x,w);

disp('15. Combining the testing features from All DCNN');
% Merge Resnet and googlenet deep features for testing
y = horzcat(resnet_features_test,googlenet_features_test);
% Merge inceptionv3 and densenet deep features for testing
z = horzcat(inceptionv3_features_test, densenet_features_test);
% Merge all deep features for testing
new_F_test = horzcat(y,z);


%Get Train Label from training dataset
train_labels = grp2idx(trainingSet.Labels);

%Get Test Label from testing dataset
test_labels = grp2idx(testingSet.Labels);

disp('16. creating training and testing dataset for elm');
%Give labels to training and testing features
 training = horzcat(train_labels,new_F_train);
 testing = horzcat( test_labels,new_F_test);


C = 2^-10;
%disp('17. Classification using ELM');
%[TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
ELM(training, testing, 1, 10000, 'sig', C);

%define Number subnetworknode for MFB_ELM
number_subnetwork_node =10;
```

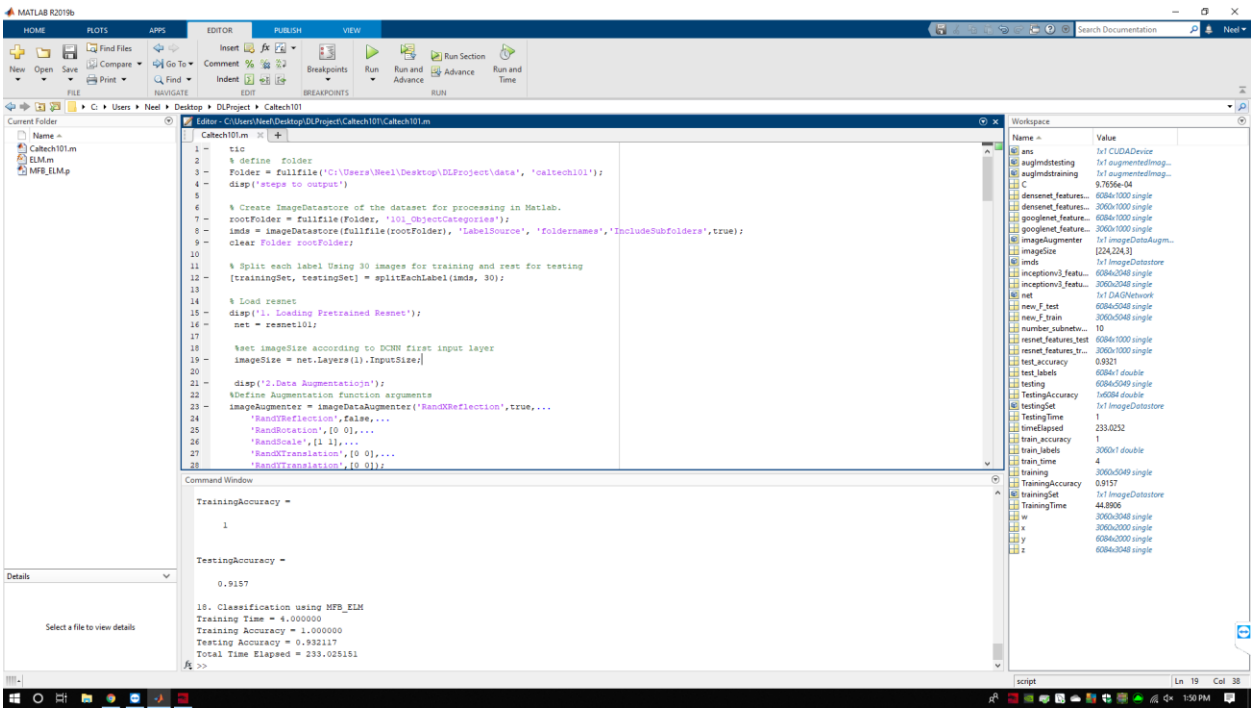
```
disp('18. Classification using MFB_ELM')
[train_time,
train_accuracy,test_accuracy]=MFB_ELM(training,testing,1,1,'sig',number_subne
twork_node,C);
fprintf('Training Time = %f\n',train_time);
fprintf('Training Accuracy = %f\n',train_accuracy);
fprintf('Testing Accuracy = %f\n',test_accuracy);



timeElapsed = toc;
```

Results:

- **Top 1-Accuracy: 93.23%**
- **Accuracy in each Trial: 93.2117%, 93.0309% and 93.2281%**
- **GPU: NVIDIA GEFORCE RTX 2080 Ti 11GB**
- **Total time: 0.1859hr = 11.1547minutes**

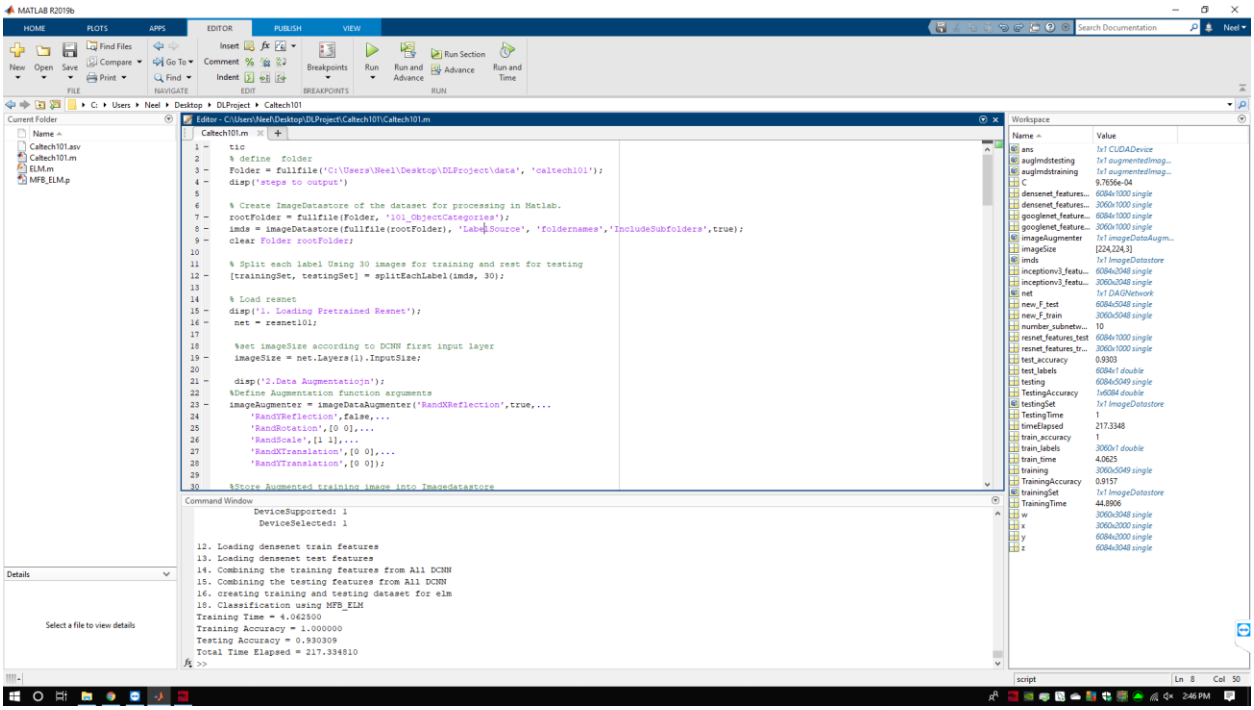
Outputs:

1.Run (Testing Acc: - 93.2117%)



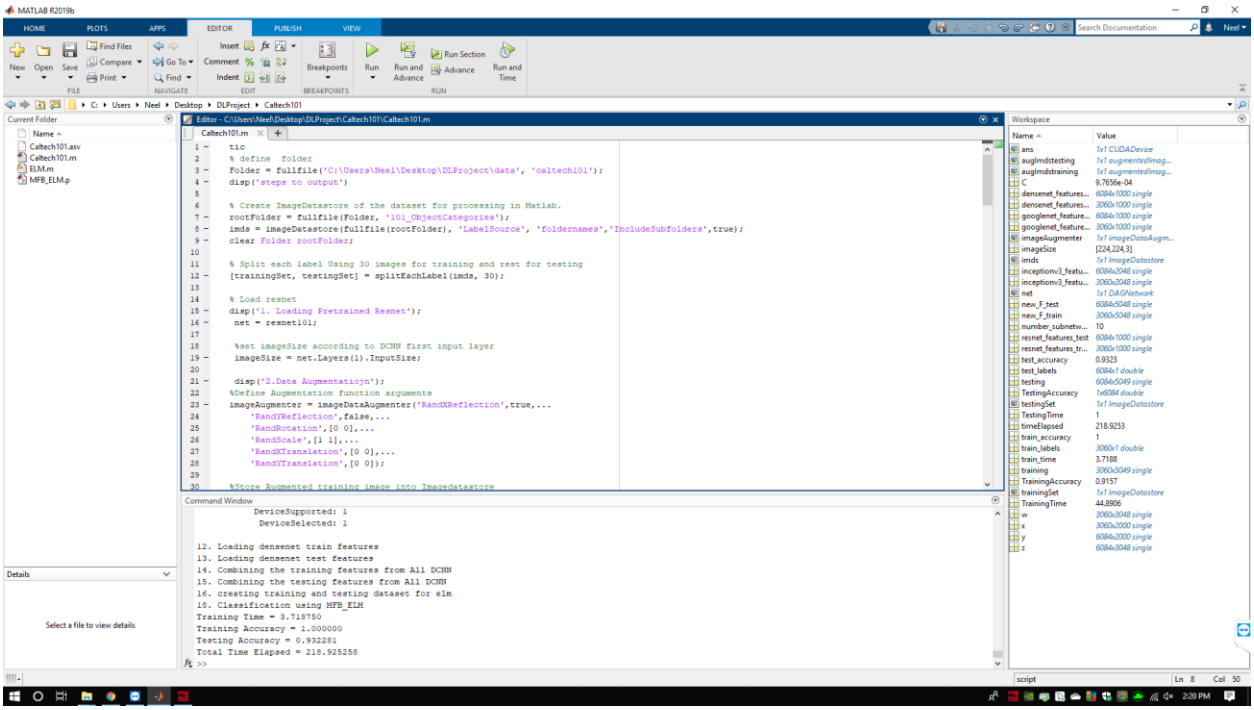

```
Command Window
                    MaxGridSize:  [2.1475e+09  65535  65535]
                      SIMDWidth:  32
                    TotalMemory:  1.1811e+10
                AvailableMemory:  9.5410e+09
            MultiprocessorCount:  68
                   ClockRateKHz:  1545000
                    ComputeMode:  'Default'
      GPUOverlapsTransfers:  1
    KernelExecutionTimeout:  1
          CanMapHostMemory:  1
           DeviceSupported:  1
            DeviceSelected:  1

12. Loading densenet train features
13. Loading densenet test features
14. Combining the training features from All DCNN
15. Combining the testing features from All DCNN
16. creating training and testing dataset for elm
17. Classification using ELM

TrainingTime =

    44.8906


TestingTime =

    10.2969


TrainingAccuracy =

     1


TestingAccuracy =

    0.9157

18. Classification using MFB_ELM
Training Time = 4.000000
Training Accuracy = 1.000000
Testing Accuracy = 0.932117
Total Time Elapsed = 233.025151
>>
```

2.Run (Testing Acc: - 93.0309%)




```
Command Window
              CanMapHostMemory:  1
               DeviceSupported:  1
                DeviceSelected:  1

9. Loading googlenet train features
10. Loading googlenet test features
11. Loading Pretrained densenet201

ans =

  CUDADevice with properties:

                         Name:  'GeForce RTX 2080 Ti'
                        Index:  1
          ComputeCapability:  '7.5'
              SupportsDouble:  1
               DriverVersion:  11
              ToolkitVersion:  10.1000
          MaxThreadsPerBlock:  1024
            MaxShmemPerBlock:  49152
          MaxThreadBlockSize:  [1024 1024 64]
                 MaxGridSize:  [2.1475e+09 65535 65535]
                   SIMDWidth:  32
                 TotalMemory:  1.1811e+10
             AvailableMemory:  9.5410e+09
        MultiprocessorCount:  68
               ClockRateKHz:  1545000
                ComputeMode:  'Default'
       GPUOverlapsTransfers:  1
     KernelExecutionTimeout:  1
           CanMapHostMemory:  1
            DeviceSupported:  1
             DeviceSelected:  1

12. Loading densenet train features
13. Loading densenet test features
14. Combining the training features from All DCNN
15. Combining the testing features from All DCNN
16. creating training and testing dataset for elm
18. Classification using MFB_ELM
Training Time = 4.062500
Training Accuracy = 1.000000
Testing Accuracy = 0.930309
Total Time Elapsed = 217.334810
>>
```

3.Run (Testing Acc: - 93.2281%)



```
Command Window

   9. Loading googlenet train features
   10. Loading googlenet test features
   11. Loading Pretrained densenet201

   ans =

     CUDADevice with properties:

                         Name: 'GeForce RTX 2080 Ti'
                        Index: 1
          ComputeCapability: '7.5'
              SupportsDouble: 1
               DriverVersion: 11
              ToolkitVersion: 10.1000
          MaxThreadsPerBlock: 1024
            MaxShmemPerBlock: 49152
          MaxThreadBlockSize: [1024 1024 64]
                 MaxGridSize: [2.1475e+09 65535 65535]
                   SIMDWidth: 32
                 TotalMemory: 1.1811e+10
             AvailableMemory: 9.5410e+09
        MultiprocessorCount: 68
               ClockRateKHz: 1545000
                ComputeMode: 'Default'
       GPUOverlapsTransfers: 1
     KernelExecutionTimeout: 1
           CanMapHostMemory: 1
            DeviceSupported: 1
             DeviceSelected: 1

   12. Loading densenet train features
   13. Loading densenet test features
   14. Combining the training features from All DCNN
   15. Combining the testing features from All DCNN
   16. creating training and testing dataset for elm
   18. Classification using MFB_ELM
   Training Time = 3.718750
   Training Accuracy = 1.000000
   Testing Accuracy = 0.932281
   Total Time Elapsed = 218.925258
fx >>
```

2. Caltech256

Code:

```
tic
% define  folder
Folder = fullfile('C:\Users\Neel\Desktop\DLProject\data', 'caltech256');
disp('steps to output')

% Create ImageDatastore of the dataset for processing in Matlab.
rootFolder = fullfile(Folder, '256_ObjectCategories');
imds = imageDatastore(fullfile(rootFolder), 'LabelSource',
'foldernames','IncludeSubfolders',true);
clear Folder rootFolder;

% Split each label Using 30 images for training and rest for testing
[trainingSet, testingSet] = splitEachLabel(imds, 30);

% Load resnet
disp('1. Loading Pretrained Resnet');
 net = resnet101;

 %set imageSize according to DCNN first input layer
 imageSize = net.Layers(1).InputSize;

 disp('2.Data Augmentatiojn');
%Define Augmentation function arguments
imageAugmenter = imageDataAugmenter('RandXReflection',true,...
    'RandYReflection',false,...
    'RandRotation',[0 0],...
    'RandScale',[1 1],...
    'RandXTranslation',[0 0],...
    'RandYTranslation',[0 0]);

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);
%resetgpu
gpuDevice(1)
```

```
% Get features from resnet
 disp('3. Loading Resnet train features');
 %Extract training features from resnet101 DCNN
 resnet_features_train =
activations(net,augImdstraining,'fc1000','MiniBatchSize',200);

 disp('4. Loading Resnet test features');
 %Extract testing features from resnet101 DCNN
 resnet_features_test =
activations(net,augImdstesting,'fc1000','MiniBatchSize',200);

 %Reshape training and testing features from resnet101
 resnet_features_train =
reshape(resnet_features_train,[1*1*1000,size(resnet_features_train,4)])' ;
 resnet_features_test =
reshape(resnet_features_test,[1*1*1000,size(resnet_features_test,4)])';

 disp('5. Loading Pretrained inceptionv3');
% Load inceptionv3
net = inceptionv3;

%set imageSize according to DCNN first input layer
imageSize = net.Layers(1).InputSize;

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb' ,'DataAugmentation',imageAugmenter);

%resetgpu
gpuDevice(1)

disp('6. Loading inceptionv3 train features');
 %Extract training features from inceptionv3 DCNN
inceptionv3_features_train =
activations(net,augImdstraining,'avg_pool','MiniBatchSize',200);

disp('7. Loading inceptionv3 test features');
 %Extract testing features from inceptionv3 DCNN
inceptionv3_features_test =
activations(net,augImdstesting,'avg_pool','MiniBatchSize',200);

%Reshape training and testing features from inceptionv3
inceptionv3_features_train =
reshape(inceptionv3_features_train,[1*1*2048,size(inceptionv3_features_train,
4)])' ;
inceptionv3_features_test =
reshape(inceptionv3_features_test,[1*1*2048,size(inceptionv3_features_test,4)
])';

disp('8. Loading Pretrained googlenet');
```

```
% Load inceptionv3
net = googlenet;

%set imageSize according to DCNN first input layer
imageSize = net.Layers(1).InputSize;

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);
%resetgpu
gpuDevice(1)

 disp('9. Loading googlenet train features');
 %Extract training features from googlenet DCNN
  googlenet_features_train = activations(net,augImdstraining,'loss3-
classifier','MiniBatchSize',200);

  disp('10. Loading googlenet test features');
  %Extract testing features from googlenet DCNN
googlenet_features_test = activations(net,augImdstesting,'loss3-
classifier','MiniBatchSize',200);

%Reshape training and testing features from googlenet
googlenet_features_train =
reshape(googlenet_features_train,[1*1*1000,size(googlenet_features_train,4)])
' ;
googlenet_features_test =
reshape(googlenet_features_test,[1*1*1000,size(googlenet_features_test,4)])';

disp('11. Loading Pretrained densenet201');
% Load densenet201
net = densenet201;

%set imageSize according to DCNN first input layer
imageSize = net.Layers(1).InputSize;

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%resetgpu
gpuDevice(1)

 disp('12. Loading densenet train features');
```

```
%Extract training features from densenet DCNN
  densenet_features_train =
activations(net,augImdstraining,'fc1000','MiniBatchSize',200);

  disp('13. Loading densenet test features');
  %Extract testing features from densenet DCNN
densenet_features_test =
activations(net,augImdstesting,'fc1000','MiniBatchSize',200);

%Reshape training and testing features from densenet
densenet_features_train =
reshape(densenet_features_train,[1*1*1000,size(densenet_features_train,4)])'
;
densenet_features_test =
reshape(densenet_features_test,[1*1*1000,size(densenet_features_test,4)])';


disp('14. Combining the training features from All DCNN');
% Merge Resnet and googlenet deep features for training
x = horzcat(resnet_features_train,googlenet_features_train);
% Merge densenet and inceptionv3 deep features for training
w = horzcat(inceptionv3_features_train, densenet_features_train);
% Merge all deep features for training
new_F_train = horzcat(x,w);

disp('15. Combining the testing features from All DCNN');
% Merge Resnet and googlenet deep features for testing
y = horzcat(resnet_features_test,googlenet_features_test);
% Merge inceptionv3 and densenet deep features for testing
z = horzcat(inceptionv3_features_test, densenet_features_test);
% Merge all deep features for testing
new_F_test = horzcat(y,z);


%Get Train Label from training dataset
train_labels = grp2idx(trainingSet.Labels);

%Get Test Label from testing dataset
test_labels = grp2idx(testingSet.Labels);

disp('16. creating training and testing dataset for elm');
%Give labels to training and testing features
 training = horzcat(train_labels,new_F_train);
 testing = horzcat( test_labels,new_F_test);


C = 2^-10;
% disp('17. Classification using ELM');
% [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
ELM(training, testing, 1, 10000, 'sig', C);

%define Number subnetworknode for MFB_ELM
number_subnetwork_node =10;
```

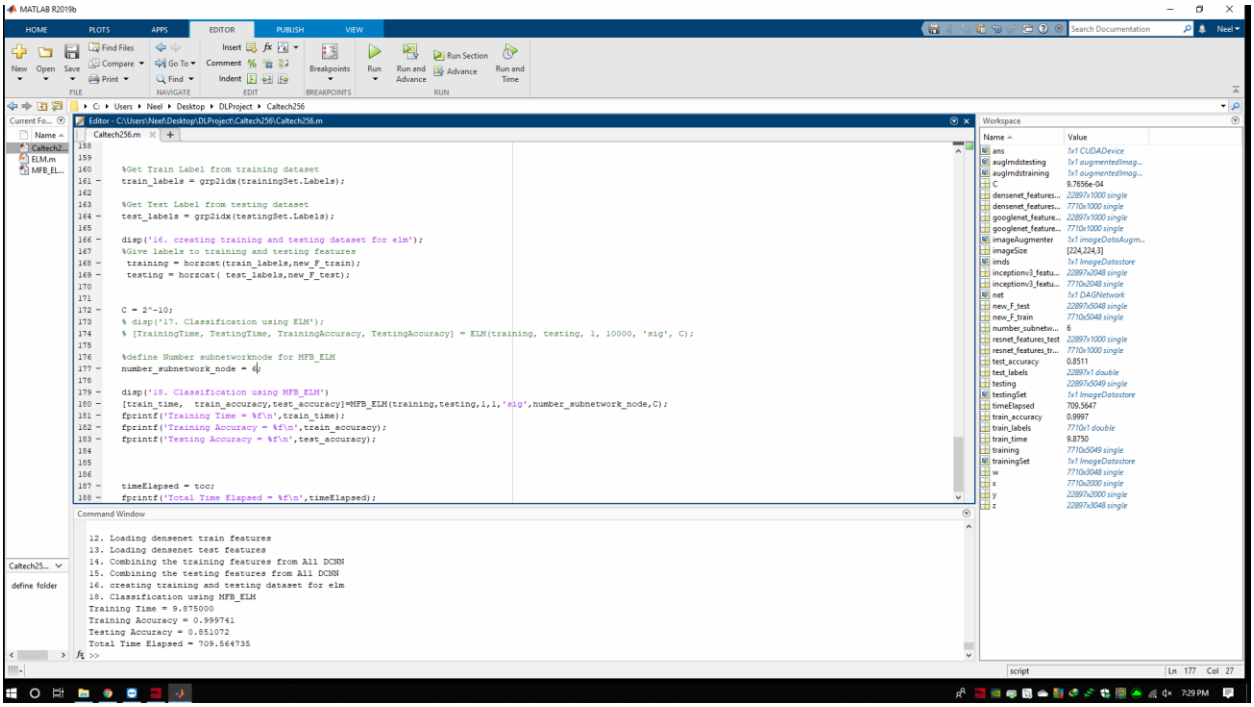
```
disp('18. Classification using MFB_ELM')
[train_time,
train_accuracy,test_accuracy]=MFB_ELM(training,testing,1,1,'sig',number_subne
twork_node,C);
fprintf('Training Time = %f\n',train_time);
fprintf('Training Accuracy = %f\n',train_accuracy);
fprintf('Testing Accuracy = %f\n',test_accuracy);



timeElapsed = toc;
```

Results:

- **Top 1-Accuracy: 85.14%**
- **Accuracy in each Trial: 85.10%, 85.04% and 85.14%**
- **GPU: NVIDIA GEFORCE RTX 2080 Ti 11Gb**
- **Total time: 0.5737hr = 34.421minutes**

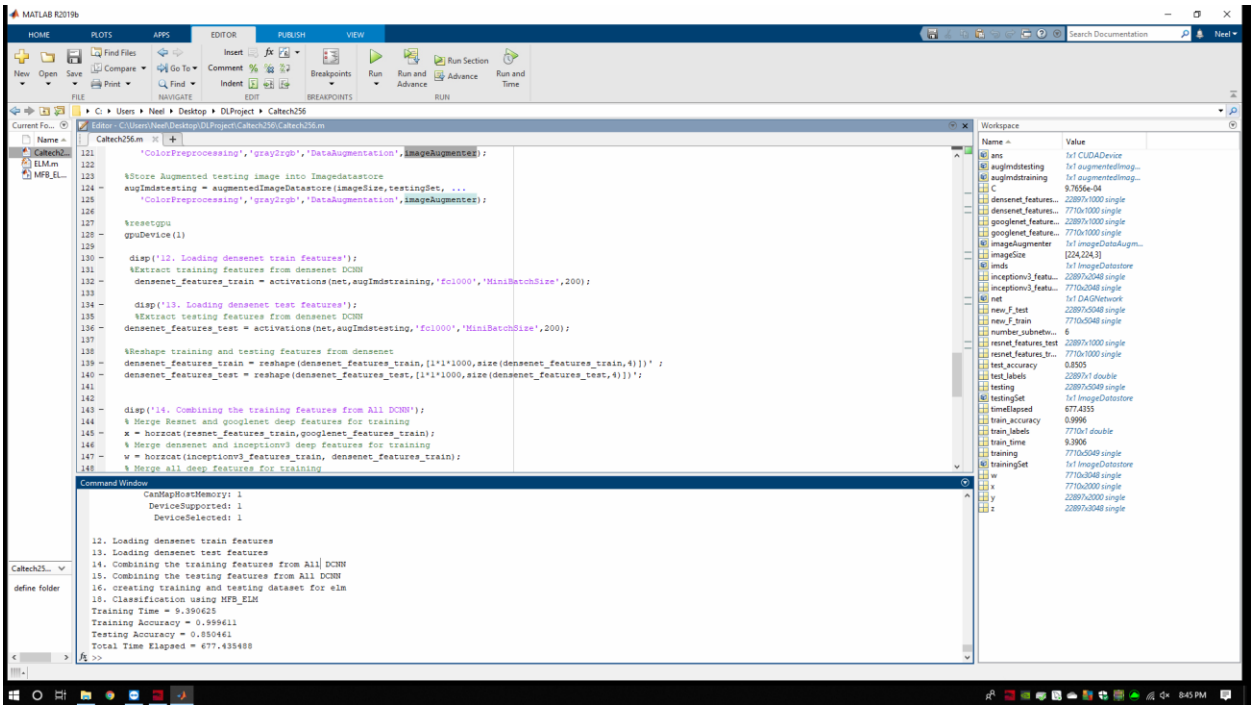
Outputs:

1.Run (Testing Acc: - 85.10%)



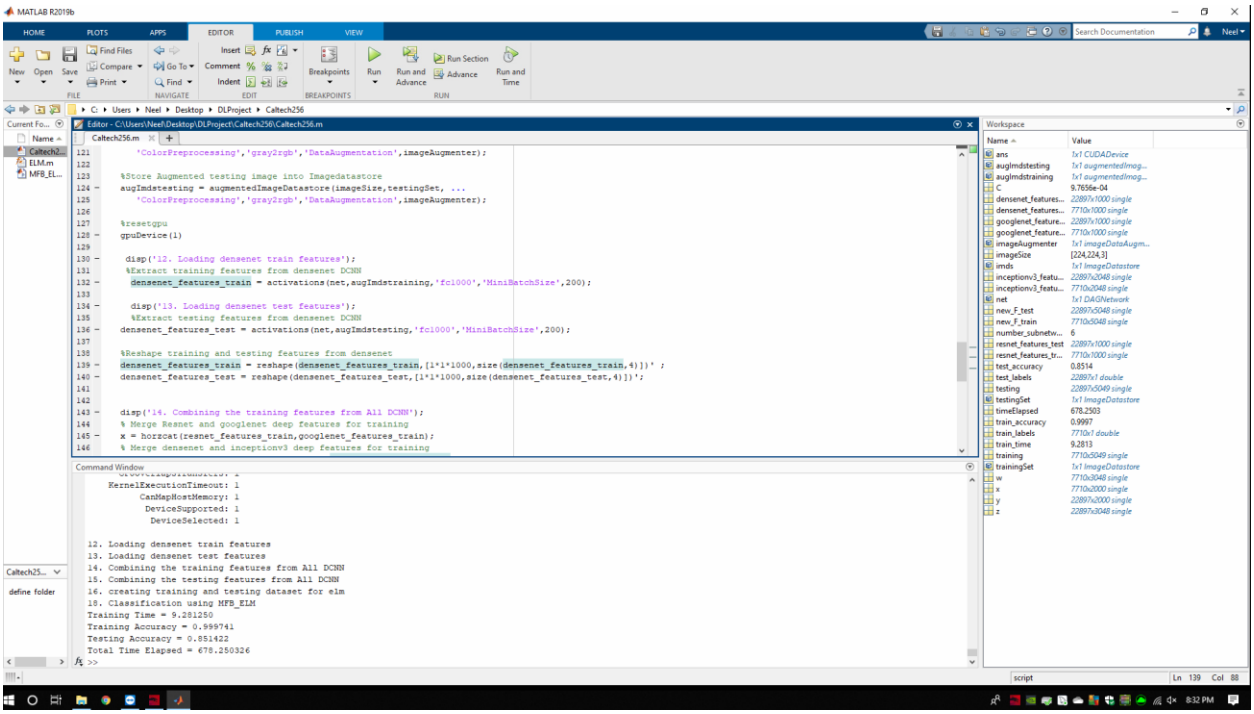

```
158
Command Window

   9. Loading googlenet train features
   10. Loading googlenet test features
   11. Loading Pretrained densenet201

   ans =

     CUDADevice with properties:

                         Name: 'GeForce RTX 2080 Ti'
                        Index: 1
          ComputeCapability: '7.5'
              SupportsDouble: 1
               DriverVersion: 11
             ToolkitVersion: 10.1000
          MaxThreadsPerBlock: 1024
            MaxShmemPerBlock: 49152
          MaxThreadBlockSize: [1024 1024 64]
                 MaxGridSize: [2.1475e+09 65535 65535]
                   SIMDWidth: 32
                 TotalMemory: 1.1811e+10
             AvailableMemory: 9.5410e+09
        MultiprocessorCount: 68
               ClockRateKHz: 1545000
                ComputeMode: 'Default'
       GPUOverlapsTransfers: 1
     KernelExecutionTimeout: 1
           CanMapHostMemory: 1
            DeviceSupported: 1
             DeviceSelected: 1

   12. Loading densenet train features
   13. Loading densenet test features
   14. Combining the training features from All DCNN
   15. Combining the testing features from All DCNN
   16. creating training and testing dataset for elm
   18. Classification using MFB_ELM
   Training Time = 9.875000
   Training Accuracy = 0.999741
   Testing Accuracy = 0.851072
   Total Time Elapsed = 709.564735
fx >>
```

2.Run (Testing Acc: - 85.04%)



```
Command Window
  11. Loading Pretrained densenet201

  ans =

    CUDADevice with properties:

                        Name: 'GeForce RTX 2080 Ti'
                       Index: 1
          ComputeCapability: '7.5'
             SupportsDouble: 1
              DriverVersion: 11
             ToolkitVersion: 10.1000
         MaxThreadsPerBlock: 1024
           MaxShmemPerBlock: 49152
         MaxThreadBlockSize: [1024 1024 64]
                MaxGridSize: [2.1475e+09 65535 65535]
                   SIMDWidth: 32
                TotalMemory: 1.1811e+10
            AvailableMemory: 9.5410e+09
       MultiprocessorCount: 68
              ClockRateKHz: 1545000
               ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1

  12. Loading densenet train features
  13. Loading densenet test features
  14. Combining the training features from All DCNN
  15. Combining the testing features from All DCNN
  16. creating training and testing dataset for elm
  18. Classification using MFB_ELM
  Training Time = 9.390625
  Training Accuracy = 0.999611
  Testing Accuracy = 0.850461
  Total Time Elapsed = 677.435488
fx >>
```

3.Run (Testing Acc: - 85.14%)



```
'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);
```

Command Window

```
9. Loading googlenet train features
10. Loading googlenet test features
11. Loading Pretrained densenet201

ans =

  CUDADevice with properties:

                      Name: 'GeForce RTX 2080 Ti'
                     Index: 1
         ComputeCapability: '7.5'
            SupportsDouble: 1
             DriverVersion: 11
            ToolkitVersion: 10.1000
        MaxThreadsPerBlock: 1024
          MaxShmemPerBlock: 49152
        MaxThreadBlockSize: [1024 1024 64]
               MaxGridSize: [2.1475e+09 65535 65535]
                 SIMDWidth: 32
               TotalMemory: 1.1811e+10
           AvailableMemory: 9.5410e+09
       MultiprocessorCount: 68
              ClockRateKHz: 1545000
               ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1

12. Loading densenet train features
13. Loading densenet test features
14. Combining the training features from All DCNN
15. Combining the testing features from All DCNN
16. creating training and testing dataset for elm
18. Classification using MFB_ELM
Training Time = 9.281250
Training Accuracy = 0.999741
Testing Accuracy = 0.851422
Total Time Elapsed = 678.250326
>>
```

3. Cifar10

Code:

```
tic
% define  folder
Folder = fullfile('C:\Users\Neel\Desktop\DLProject\data\CIFAR10', 'cifar');
disp('steps to output')

% Create ImageDatastore of the dataset for processing in Matlab.
trainFolder = fullfile(Folder, 'TRAINSET');
testFolder =  fullfile(Folder, 'TESTSET');
clear Folder;

% Split  training and  testing dataset
trainingSet = imageDatastore(fullfile(trainFolder), 'LabelSource',
'foldernames','IncludeSubfolders',true);
testingSet = imageDatastore(fullfile(testFolder), 'LabelSource',
'foldernames','IncludeSubfolders',true);

% Load resnet
disp('1. Loading Pretrained Resnet');
 net = resnet101;

 %set imageSize according to DCNN first input layer
 imageSize = net.Layers(1).InputSize;

 disp('2.Data Augmentatiojn');
%Define Augmentation function arguments
imageAugmenter = imageDataAugmenter('RandXReflection',true,...
    'RandYReflection',false,...
    'RandRotation',[0 0],...
    'RandScale',[1 1],...
    'RandXTranslation',[0 0],...
    'RandYTranslation',[0 0]);

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);
%resetgpu
gpuDevice(1)
```

```
% Get features from resnet
 disp('3. Loading Resnet train features');
 %Extract training features from resnet101 DCNN
 resnet_features_train =
activations(net,augImdstraining,'fc1000','MiniBatchSize',200);

 disp('4. Loading Resnet test features');
 %Extract testing features from resnet101 DCNN
 resnet_features_test =
activations(net,augImdstesting,'fc1000','MiniBatchSize',200);

 %Reshape training and testing features from resnet101
 resnet_features_train =
reshape(resnet_features_train,[1*1*1000,size(resnet_features_train,4)])' ;
 resnet_features_test =
reshape(resnet_features_test,[1*1*1000,size(resnet_features_test,4)])';

 disp('5. Loading Pretrained inceptionv3');
% Load inceptionv3
net = inceptionv3;

%set imageSize according to DCNN first input layer
imageSize = net.Layers(1).InputSize;

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb' ,'DataAugmentation',imageAugmenter);

%resetgpu
gpuDevice(1)

disp('6. Loading inceptionv3 train features');
 %Extract training features from inceptionv3 DCNN
inceptionv3_features_train =
activations(net,augImdstraining,'avg_pool','MiniBatchSize',200);

disp('7. Loading inceptionv3 test features');
 %Extract testing features from inceptionv3 DCNN
inceptionv3_features_test =
activations(net,augImdstesting,'avg_pool','MiniBatchSize',200);

%Reshape training and testing features from inceptionv3
inceptionv3_features_train =
reshape(inceptionv3_features_train,[1*1*2048,size(inceptionv3_features_train,
4)])' ;
inceptionv3_features_test =
reshape(inceptionv3_features_test,[1*1*2048,size(inceptionv3_features_test,4)
])';

disp('8. Loading Pretrained googlenet');
```

```
% Load inceptionv3
net = googlenet;

%set imageSize according to DCNN first input layer
imageSize = net.Layers(1).InputSize;

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);
%resetgpu
gpuDevice(1)

 disp('9. Loading googlenet train features');
 %Extract training features from googlenet DCNN
  googlenet_features_train = activations(net,augImdstraining,'loss3-
classifier','MiniBatchSize',200);

  disp('10. Loading googlenet test features');
  %Extract testing features from googlenet DCNN
googlenet_features_test = activations(net,augImdstesting,'loss3-
classifier','MiniBatchSize',200);

%Reshape training and testing features from googlenet
googlenet_features_train =
reshape(googlenet_features_train,[1*1*1000,size(googlenet_features_train,4)])
' ;
googlenet_features_test =
reshape(googlenet_features_test,[1*1*1000,size(googlenet_features_test,4)])';

disp('11. Loading Pretrained densenet201');
% Load densenet201
net = densenet201;

%set imageSize according to DCNN first input layer
imageSize = net.Layers(1).InputSize;

%Store Augmented training image into Imagedatastore
augImdstraining = augmentedImageDatastore(imageSize,trainingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%Store Augmented testing image into Imagedatastore
augImdstesting = augmentedImageDatastore(imageSize,testingSet, ...
   'ColorPreprocessing','gray2rgb','DataAugmentation',imageAugmenter);

%resetgpu
gpuDevice(1)

 disp('12. Loading densenet train features');
```

```
%Extract training features from densenet DCNN
  densenet_features_train =
activations(net,augImdstraining,'fc1000','MiniBatchSize',200);

  disp('13. Loading densenet test features');
  %Extract testing features from densenet DCNN
densenet_features_test =
activations(net,augImdstesting,'fc1000','MiniBatchSize',200);

%Reshape training and testing features from densenet
densenet_features_train =
reshape(densenet_features_train,[1*1*1000,size(densenet_features_train,4)])'
;
densenet_features_test =
reshape(densenet_features_test,[1*1*1000,size(densenet_features_test,4)])';


disp('14. Combining the training features from All DCNN');
% Merge Resnet and googlenet deep features for training
x = horzcat(resnet_features_train,googlenet_features_train);
% Merge densenet and inceptionv3 deep features for training
w = horzcat(inceptionv3_features_train, densenet_features_train);
% Merge all deep features for training
new_F_train = horzcat(x,w);

disp('15. Combining the testing features from All DCNN');
% Merge Resnet and googlenet deep features for testing
y = horzcat(resnet_features_test,googlenet_features_test);
% Merge inceptionv3 and densenet deep features for testing
z = horzcat(inceptionv3_features_test, densenet_features_test);
% Merge all deep features for testing
new_F_test = horzcat(y,z);


%Get Train Label from training dataset
train_labels = grp2idx(trainingSet.Labels);

%Get Test Label from testing dataset
test_labels = grp2idx(testingSet.Labels);

disp('16. creating training and testing dataset for elm');
%Give labels to training and testing features
 training = horzcat(train_labels,new_F_train);
 testing = horzcat( test_labels,new_F_test);


C = 2^-10;
disp('17. Classification using ELM');
[TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
ELM(training, testing, 1, 10000, 'sig', C);

%define Number subnetworknode for MFB_ELM
number_subnetwork_node =10;
```

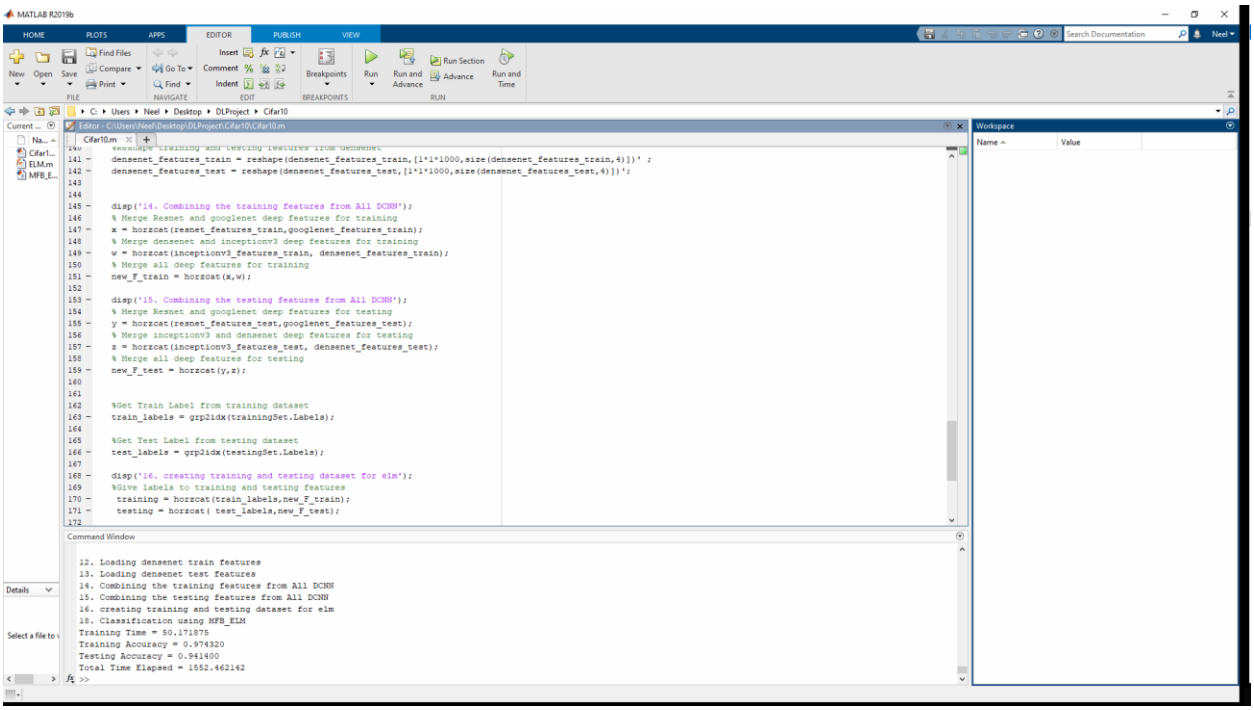
```
disp('18. Classification using MFB_ELM')
[train_time,
train_accuracy,test_accuracy]=MFB_ELM(training,testing,1,1,'sig',number_subne
twork_node,C);
fprintf('Training Time = %f\n',train_time);
fprintf('Training Accuracy = %f\n',train_accuracy);
fprintf('Testing Accuracy = %f\n',test_accuracy);



timeElapsed = toc;
```

Results:

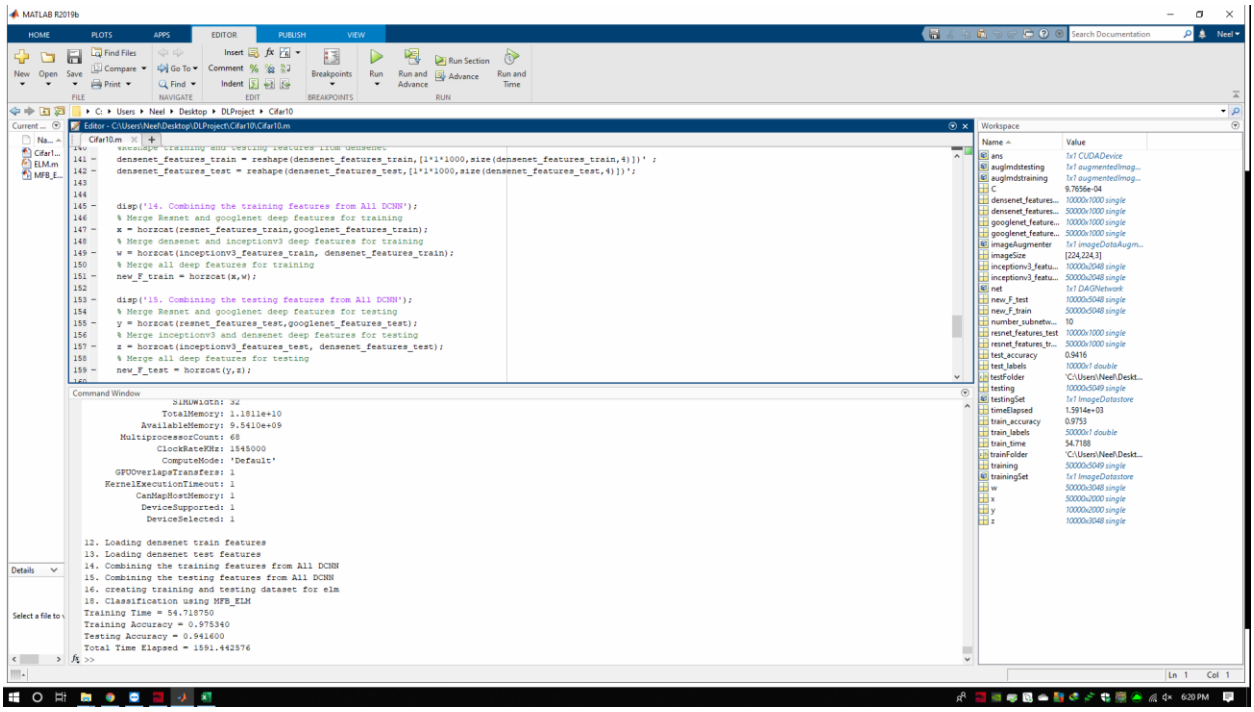
- **Top 1-Accuracy: 94.16%**
- **Accuracy in each Trial: 94.14%, 94.16% and 94.15%**
- **GPU: NVIDIA GEFORCE RTX 2080 Ti 11Gb**
- **Total time: 1.35hr = 81.032minutes**

Outputs:

1.Run (Testing Acc: - 94.14%)



2.Run (Testing Acc: - 94.16%)

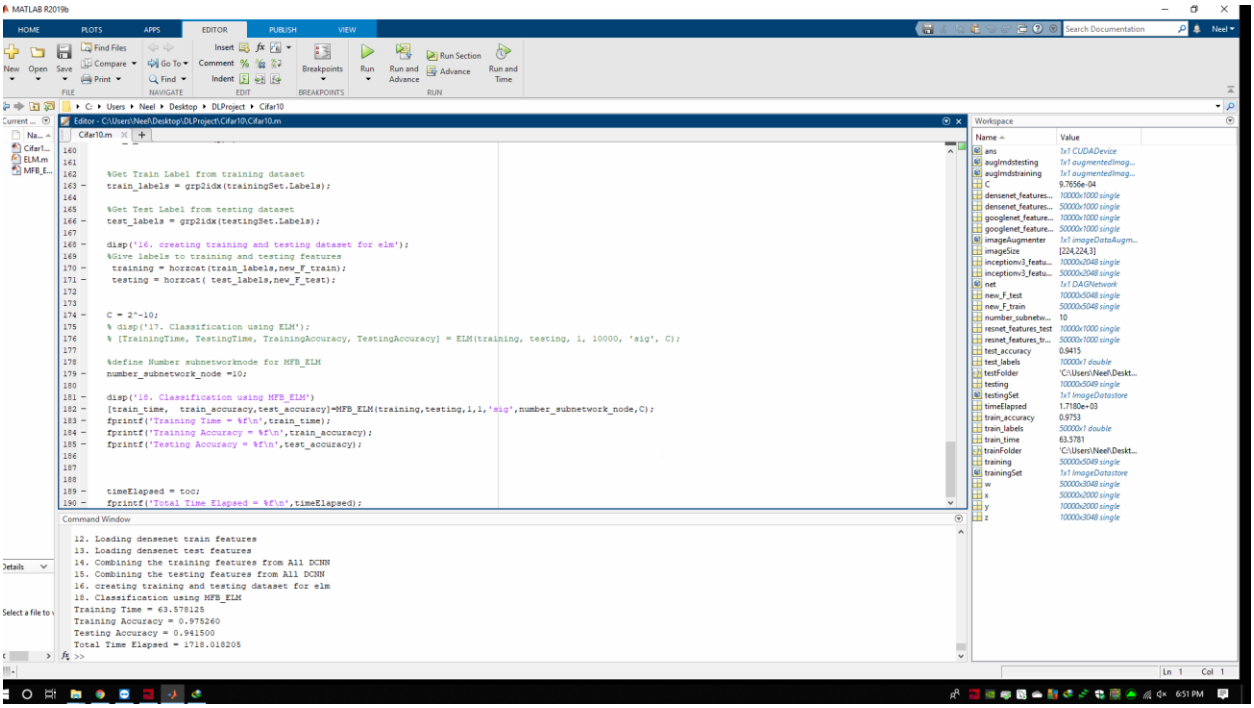


Cifar10.m

Command Window

```
              DeviceSelected: 1

  9. Loading googlenet train features
  10. Loading googlenet test features
  11. Loading Pretrained densenet201

  ans =

    CUDADevice with properties:

                          Name: 'GeForce RTX 2080 Ti'
                         Index: 1
          ComputeCapability: '7.5'
             SupportsDouble: 1
              DriverVersion: 11
             ToolkitVersion: 10.1000
          MaxThreadsPerBlock: 1024
            MaxShmemPerBlock: 49152
          MaxThreadBlockSize: [1024 1024 64]
                 MaxGridSize: [2.1475e+09 65535 65535]
                   SIMDWidth: 32
                 TotalMemory: 1.1811e+10
             AvailableMemory: 9.5410e+09
         MultiprocessorCount: 68
                ClockRateKHz: 1545000
                 ComputeMode: 'Default'
        GPUOverlapsTransfers: 1
      KernelExecutionTimeout: 1
             CanMapHostMemory: 1
              DeviceSupported: 1
              DeviceSelected: 1

  12. Loading densenet train features
  13. Loading densenet test features
  14. Combining the training features from All DCNN
  15. Combining the testing features from All DCNN
  16. creating training and testing dataset for elm
  18. Classification using MFB_ELM
  Training Time = 54.718750
  Training Accuracy = 0.975340
  Testing Accuracy = 0.941600
  Total Time Elapsed = 1591.442576
fx >>
```

3.Run (Testing Acc: - 94.15%)



Command Window

```
9. Loading googlenet train features
10. Loading googlenet test features
11. Loading Pretrained densenet201

ans =

  CUDADevice with properties:

                      Name: 'GeForce RTX 2080 Ti'
                     Index: 1
      ComputeCapability: '7.5'
          SupportsDouble: 1
           DriverVersion: 11
          ToolkitVersion: 10.1000
    MaxThreadsPerBlock: 1024
      MaxShmemPerBlock: 49152
    MaxThreadBlockSize: [1024 1024 64]
             MaxGridSize: [2.1475e+09 65535 65535]
               SIMDWidth: 32
             TotalMemory: 1.1811e+10
         AvailableMemory: 9.5410e+09
    MultiprocessorCount: 68
           ClockRateKHz: 1545000
            ComputeMode: 'Default'
   GPUOverlapsTransfers: 1
KernelExecutionTimeout: 1
       CanMapHostMemory: 1
        DeviceSupported: 1
         DeviceSelected: 1

12. Loading densenet train features
13. Loading densenet test features
14. Combining the training features from All DCNN
15. Combining the testing features from All DCNN
16. creating training and testing dataset for elm
18. Classification using MFB_ELM
Training Time = 63.578125
Training Accuracy = 0.975260
Testing Accuracy = 0.941500
Total Time Elapsed = 1718.018205
>>
```

Average Accuracy:

- **Caltech101+Caltech256+Cifar10/3 = (93.23+85.14+94.16)/3 ≈ 91**

Reference:

- **Yimin Yang, Q.M.Jonathan Wu. Autoencoder with invertible functions for dimension reduction and image reconstruction. IEEE Transactions on Neural Networks and Learning Systems. 2018.- MFB_ELM**
- http://www.yiminyang.com/code.html
- **All Together Now! The Benefits of Adaptively Fusing Pre-trained Deep Representations. Yehezkel S. Resheff1, Itay Lieder2 and Tom Hope2.**