



Masters in Computer Science

Topics in Deep Learning

(COMP-5413)

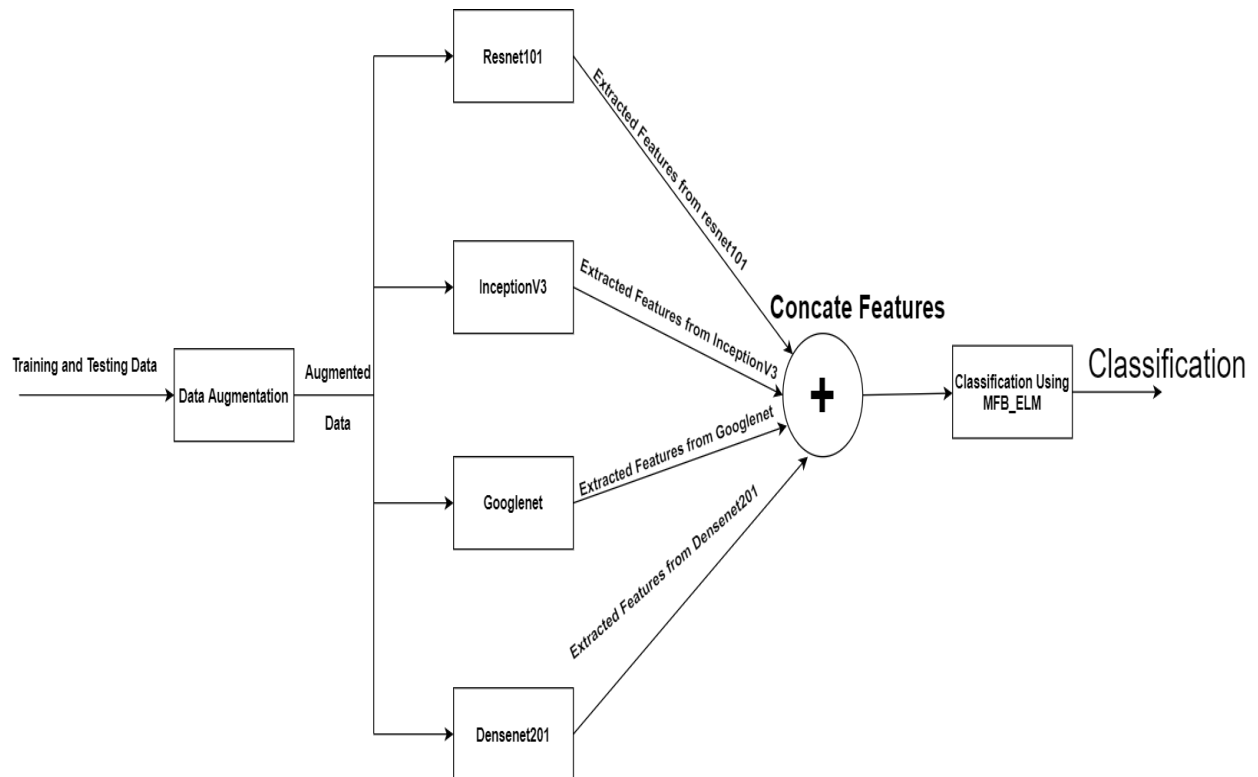
Group:

- 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
augImdsttraining = 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,augImdsttraining, '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
augImdsttraining = 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,augImdsttraining, '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
augImdsttraining = 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,augImdsttraining,'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
augImdsttraining = 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,augImdsttraining,'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);

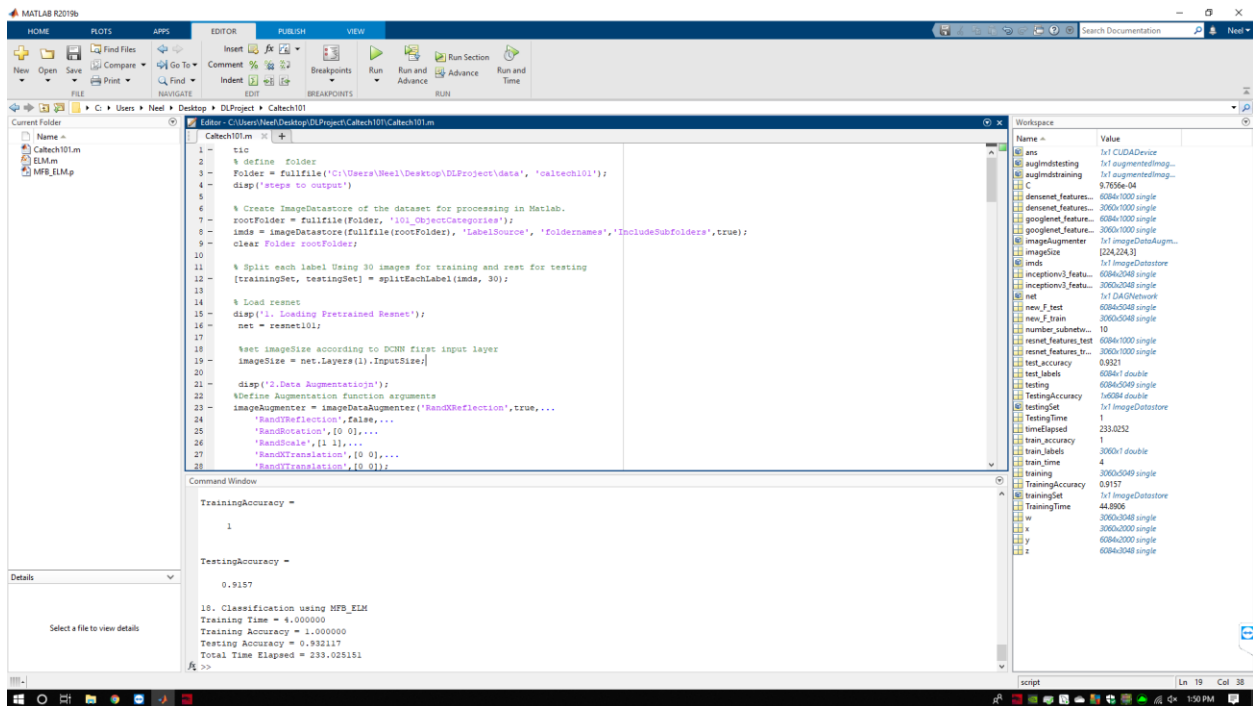
timeElapsed = toc;
```

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%)



2.Run (Testing Acc: - 93.0309%)

The MATLAB R2019b interface displays a script for image classification. The script defines a folder, creates an ImageDatastore, splits data into training and testing sets, loads pretrained Resnet features, and performs data augmentation. The Command Window shows the execution progress, including loading features, creating datasets, and final training and testing accuracies.

```

1 =
2 % define folder
3 Folder = fullfile('C:\Users\Neel\Desktop\DLProject\data', 'caltech101');
4 disp('steps to output')
5
6 % Create ImageDatastore of the dataset for processing in Matlab.
7 rootFolder = fullfile(Folder, '101_ObjectCategories');
8 imds = ImageDatastore(fullfile(rootFolder, 'LabelSource', 'foldernames', 'IncludeSubfolders', true);
9 clear Folder rootFolder;
10
11 % Split each label Using 30 images for training and rest for testing
12 [trainingSet, testingSet] = splitEachLabel(imds, 30);
13
14 % Load resnet
15 disp('1. Loading Pretrained Resnet');
16 net = resnet101;
17
18 %set imageSize according to DCNN first input layer
19 imageSize = net.Layers(1).InputSize;
20
21 %
22 disp('2. Data Augmentation');
23 %define Augmentation function arguments
24 imageAugmenter = imageAugmenter('RandReflection', true, ...
25 'RandRotation', [0 0], ...
26 'RandScale', [1 1], ...
27 'RandTranslation', [0 0], ...
28 'RandTranslation', [0 0]);
29
30 %Store Augmented training image into ImageDatastore

```

Command Window

```

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

```

Workspace

Name	Value
ans	1x1 CUDADevice
augmentedtesting	1x1 augmentedImageDatastore
augmentedtraining	1x1 augmentedImageDatastore
C	9.7556e+04
densenet_features...	6084x1000 single
densenet_features...	3000x1000 single
googlenet_features...	6084x1000 single
googlenet_features...	3000x1000 single
imageAugmenter	1x1 imageDataAugmenter
imageSize	[224 224 3]
imds	1x1 ImageDatastore
inception3_features...	6084x2048 single
inception3_features...	3000x2048 single
net	1x1 D40Network
new_F_test	6084x5048 single
new_F_train	3000x5048 single
number_subnets...	10
resnet_features_test	6084x1000 single
resnet_features_train	3000x1000 single
test_labels	6084x1 double
testing	6084x5048 single
TestingAccuracy	1x1 double
testingSet	1x1 ImageDatastore
TestingTime	1
timeElapsed	217.3348
train_accuracy	1
train_labels	3000x1 double
train_time	4.0625
training	3000x5048 single
TrainingAccuracy	0.9157
trainingSet	1x1 ImageDatastore
TrainingTime	44.8906
x	3000x1000 single
y	6084x2000 single
z	6084x5048 single

Command Window

```

CanMapHostMemory: 1
DeviceSupported: 1
DeviceSelected: 1

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

ans =

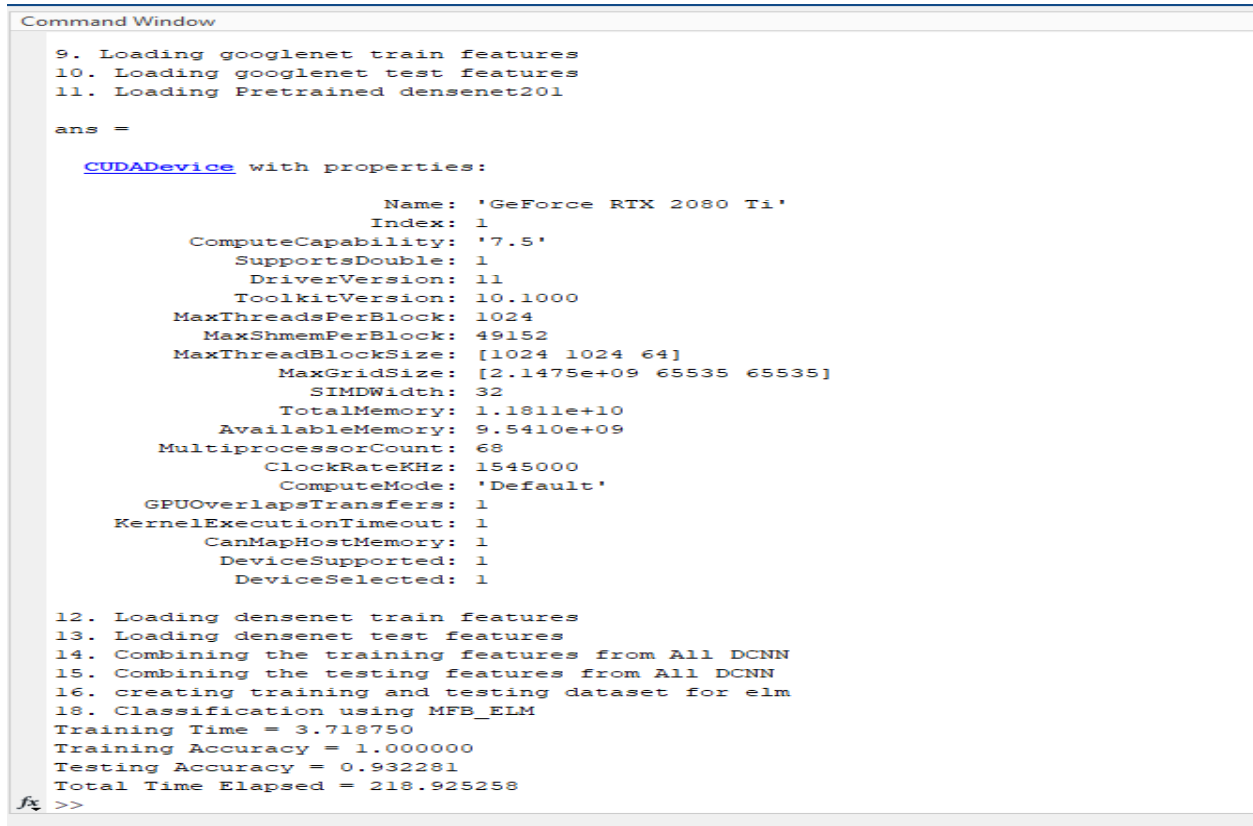
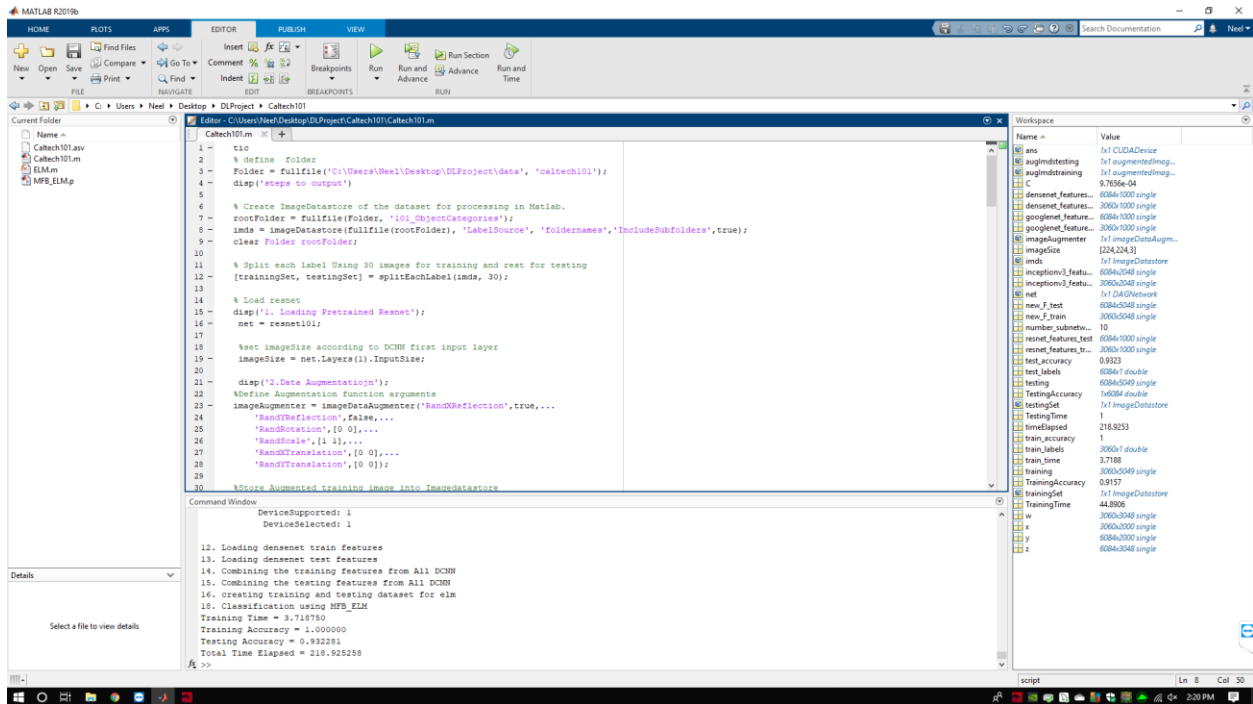
CUDADevice with properties:

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%)



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
augImdsttraining = 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,augImdsttraining,'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
augImdsttraining = 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,augImdsttraining,'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
augImdsttraining = 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,augImdsttraining,'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
augImdsttraining = 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,augImdstesting,'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);

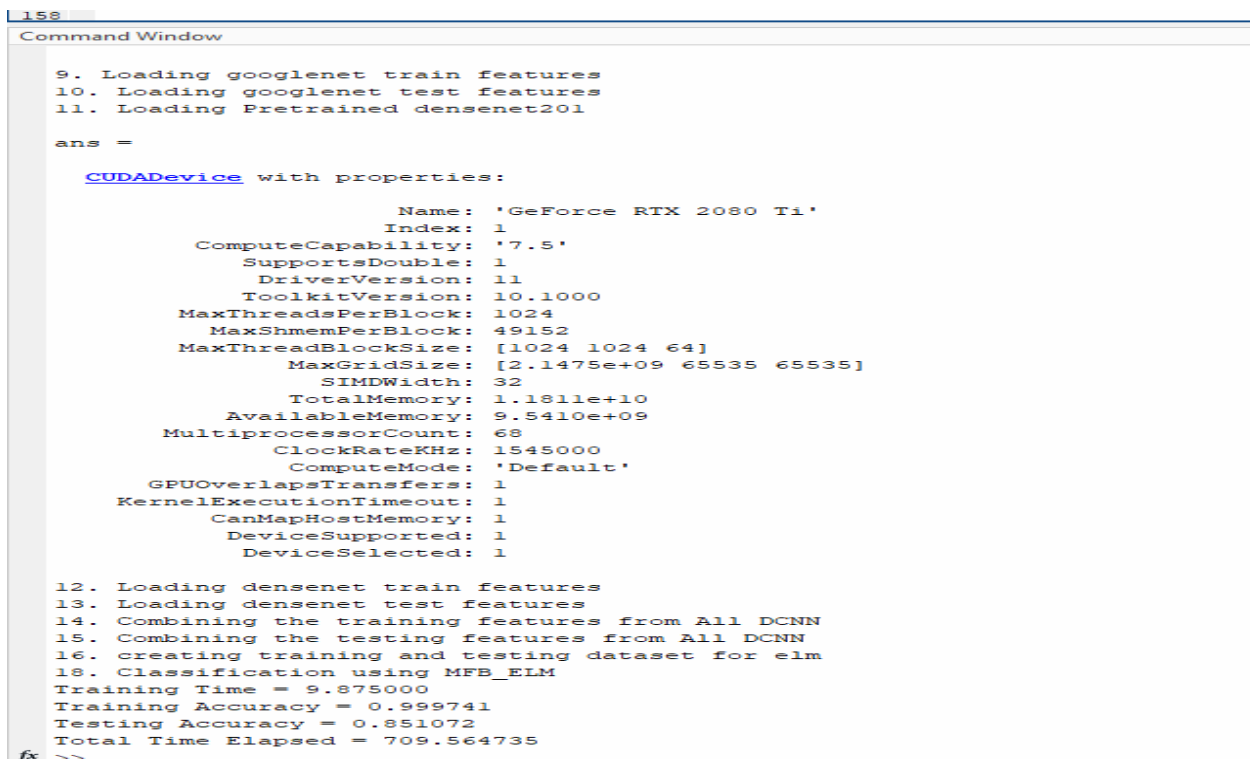
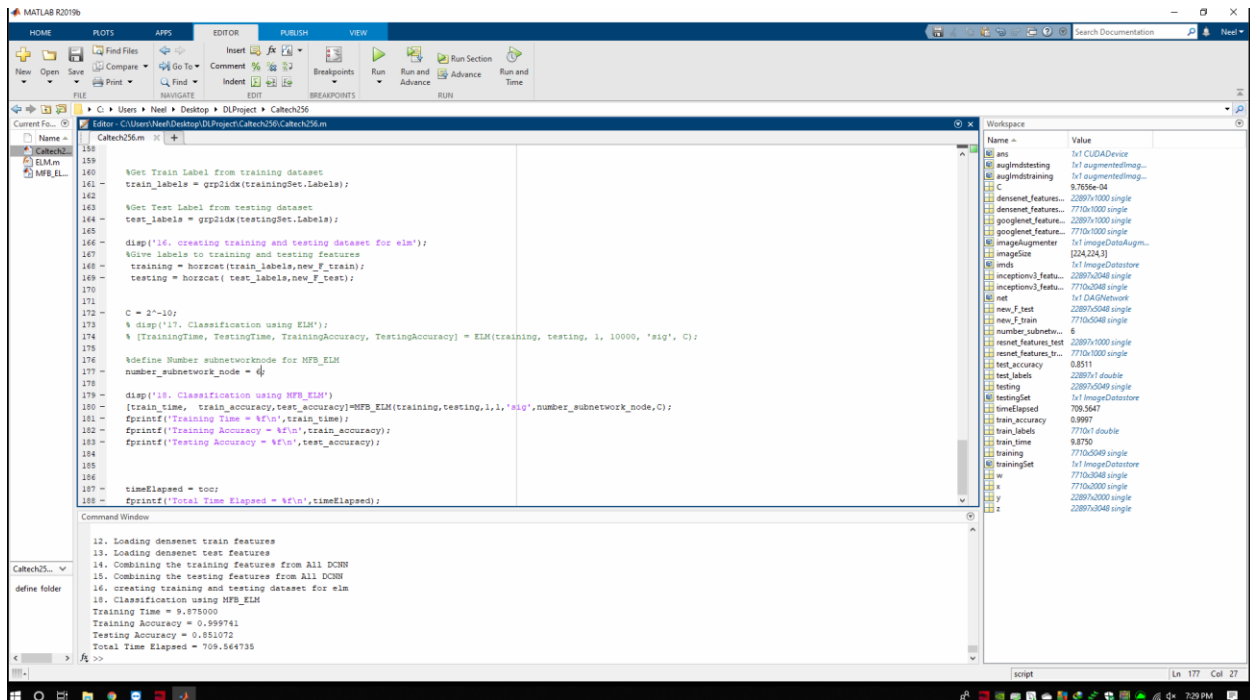
timeElapsed = toc;
```


Results:

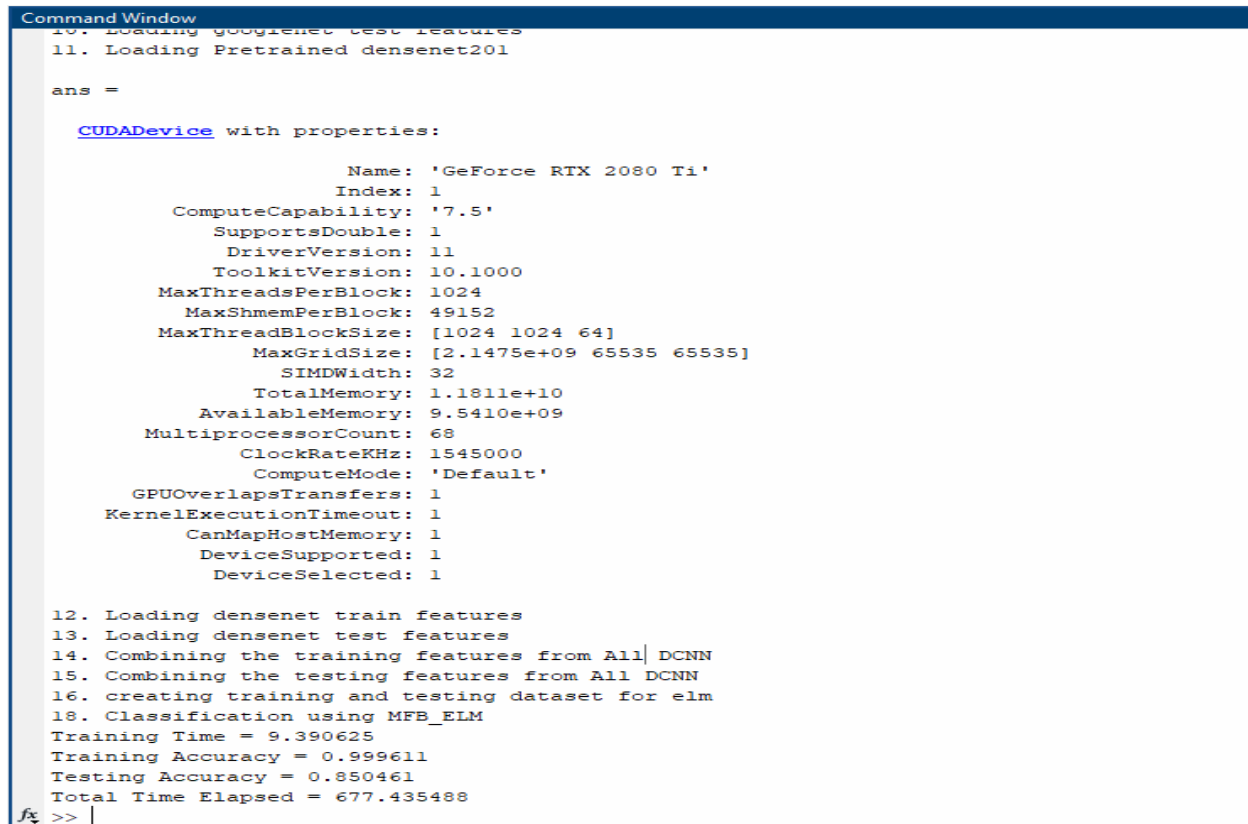
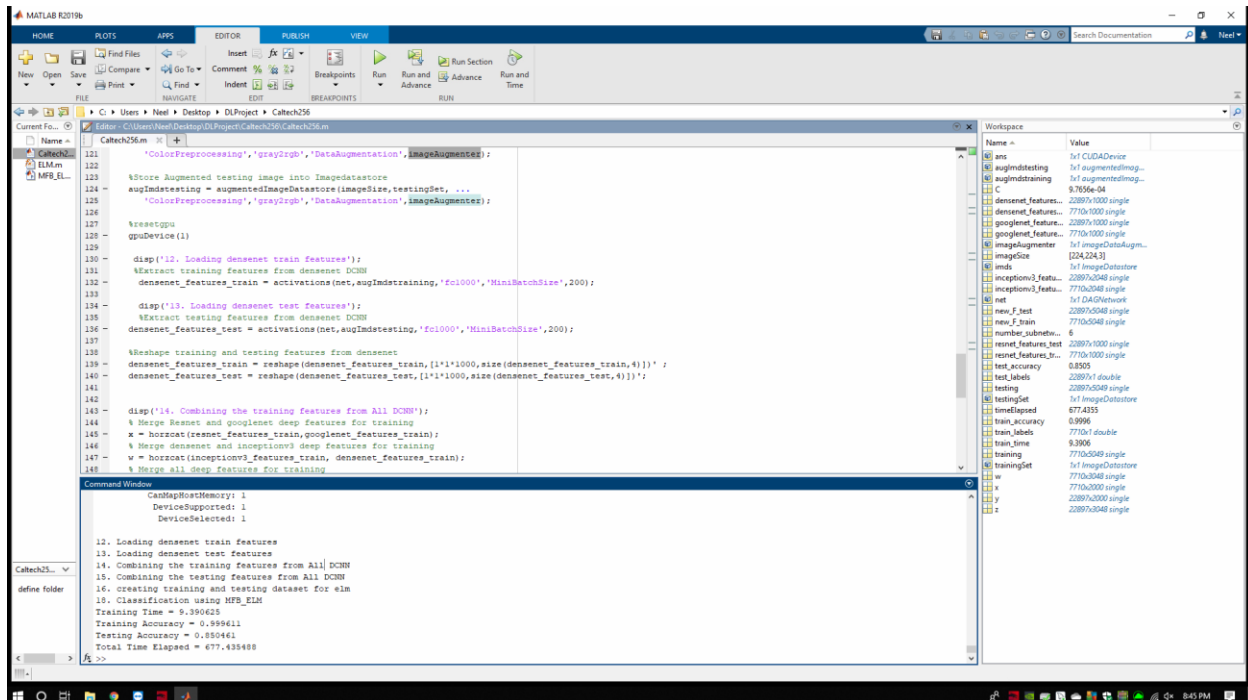
- **Top 1-Accuracy: 85.14%**
- **Accuracy in each Trial: 85.10%, 85.04% and 85.14%**
- **GPU: NVIDIA GEFORCE RTX 2080 Ti 11Gb**
- **Total time: 0.5737hr = 34.421minutes**

Outputs:

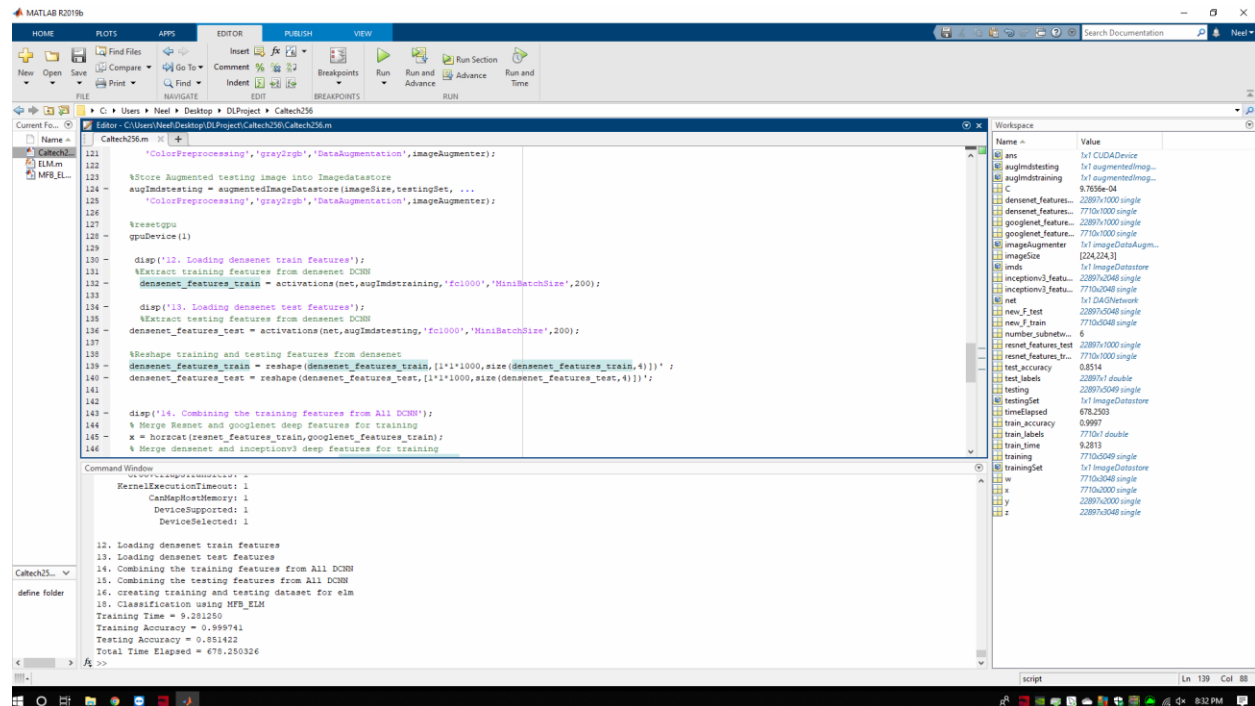
1.Run (Testing Acc: - 85.10%)



2.Run (Testing Acc: - 85.04%)



3.Run (Testing Acc: - 85.14%)



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
augImdsttraining = 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,augImdsttraining,'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
augImdsttraining = 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,augImdsttraining,'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
augImdsttraining = 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,augImdsttraining,'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
augImdsttraining = 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,augImdstesting,'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);
```

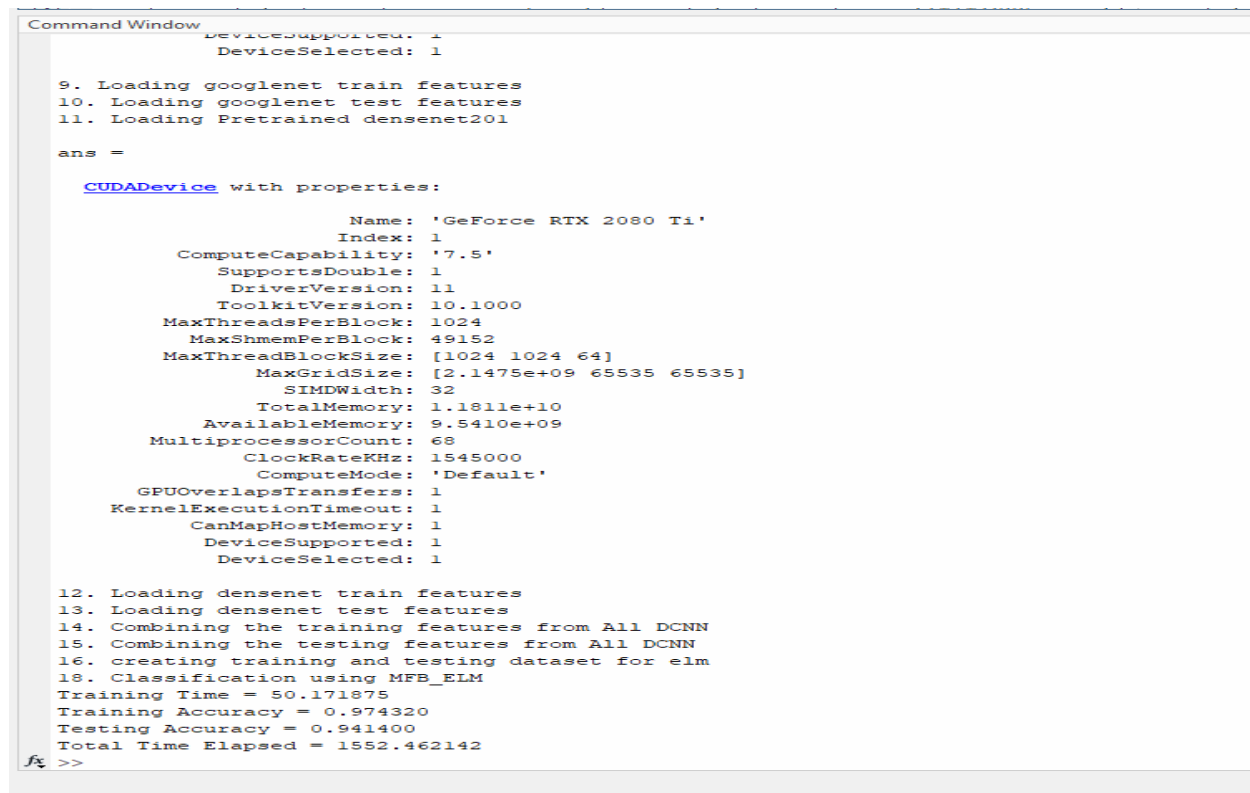
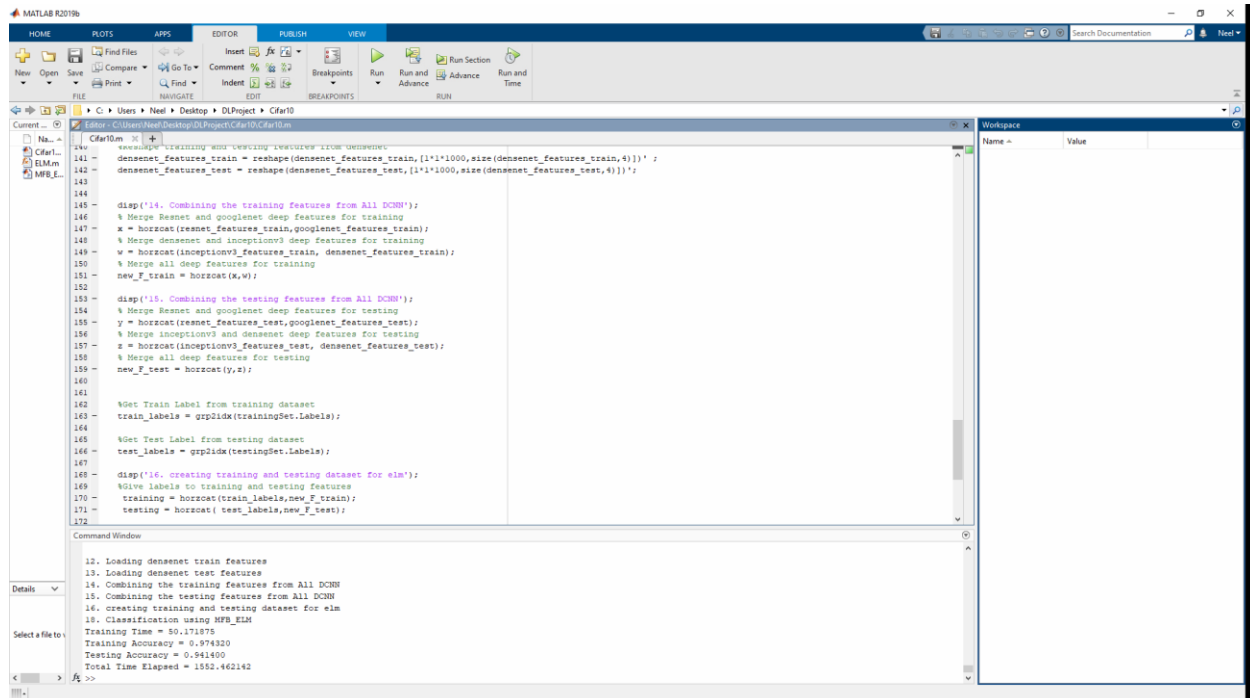
```
timeElapsed = toc;
```

Results:

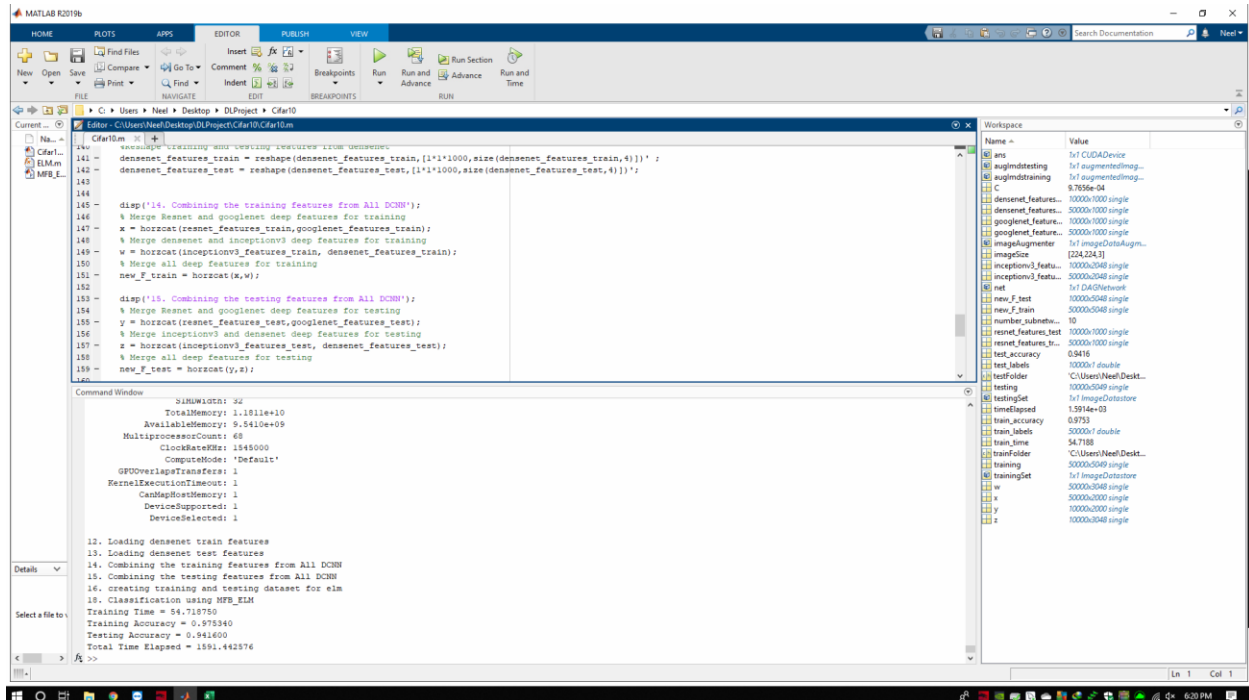
- **Top 1-Accuracy: 94.16%**
- **Accuracy in each Trial: 94.14%, 94.16% and 94.15%**
- **GPU: NVIDIA GEFORCE RTX 2080 Ti 11Gb**
- **Total time: 1.35hr = 81.032minutes**

Outputs:

1.Run (Testing Acc: - 94.14%)

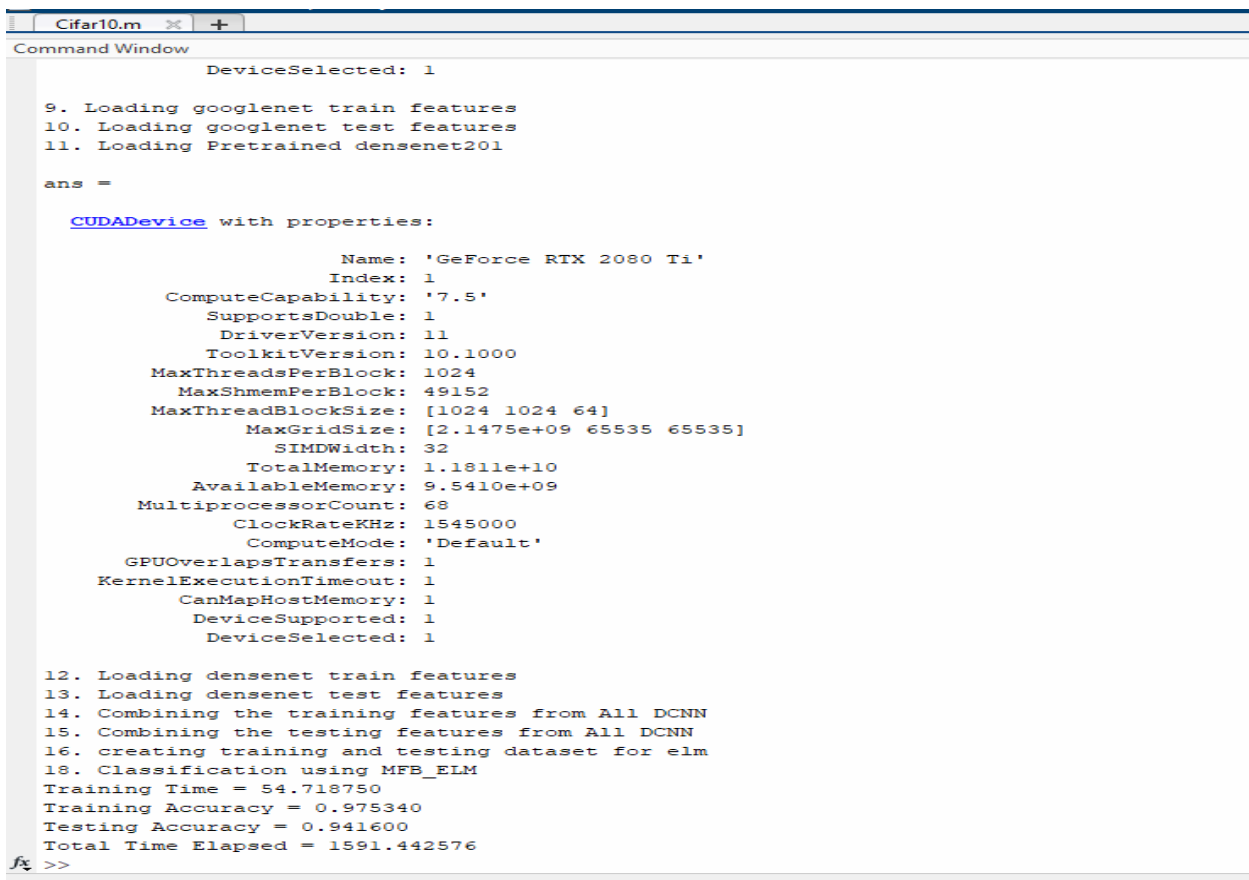


2.Run (Testing Acc: - 94.16%)



The screenshot shows the MATLAB R2019b interface. The Editor window displays the Cifar10.m script, which includes comments and code for loading, combining, and classifying features from various CNN models. The Command Window shows the execution progress, including the loading of features, combining of training and testing features, and the final classification results. The Workspace window lists the variables created during the execution, such as ans, augmentedtesting, augmentedtraining, densenet_features, googlenet_features, imageaugmented, inception3_features, new_F_train, new_F_test, new_F_train, new_F_test, number_subnets, resnet_features_test, test_accuracy, test_labels, test_folder, testing, testingSet, timeElapsed, train_accuracy, train_labels, train_time, trainFolder, training, trainingSet, w, x, y, and z.

```
141 % resnetnet_training_and_testing_features_from_DCNN
142 densenet_features_train = reshape(densenet_features_train, [1*1*1000, size(densenet_features_train, 4)]);
143 densenet_features_test = reshape(densenet_features_test, [1*1*1000, size(densenet_features_test, 4)]);
144
145 disp('14. Combining the training features from All DCNN');
146 % Merge Resnet and googlenet deep features for training
147 x = horzcat(resnet_features_train, googlenet_features_train);
148 % Merge densenet and inceptionV3 deep features for training
149 w = horzcat(inceptionV3_features_train, densenet_features_train);
150 % Merge all deep features for training
151 new_F_train = horzcat(x,w);
152
153 disp('15. Combining the testing features from All DCNN');
154 % Merge Resnet and googlenet deep features for testing
155 y = horzcat(resnet_features_test, googlenet_features_test);
156 % Merge inceptionV3 and densenet deep features for testing
157 z = horzcat(inceptionV3_features_test, densenet_features_test);
158 % Merge all deep features for testing
159 new_F_test = horzcat(y,z);
160
161 % Classification using HFB_ELM
162 Training Time = 54.718750
163 Training Accuracy = 0.975340
164 Testing Accuracy = 0.941600
165 Total Time Elapsed = 1591.442576
```



The screenshot shows the MATLAB Command Window displaying the execution of the Cifar10.m script. The output includes the device selected (GeForce RTX 2080 Ti) and the final classification results.

```
DeviceSelected: 1

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

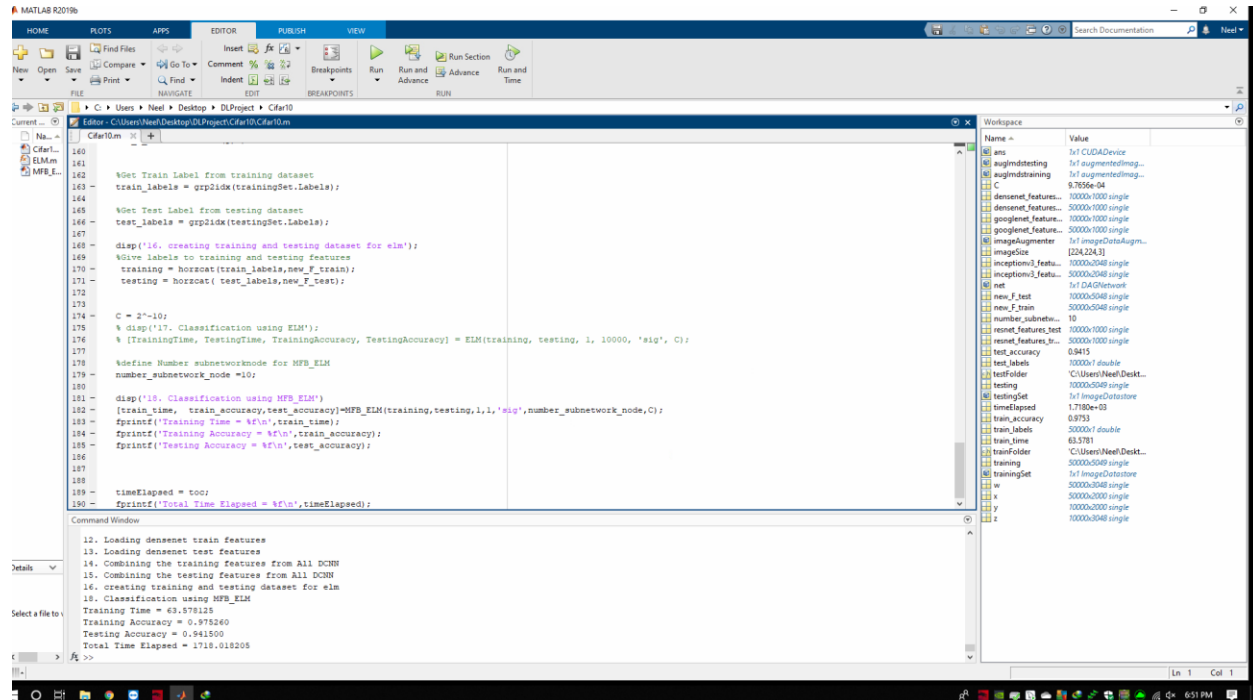
ans =

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 HFB_ELM
Training Time = 54.718750
Training Accuracy = 0.975340
Testing Accuracy = 0.941600
Total Time Elapsed = 1591.442576
>>
```

3.Run (Testing Acc: - 94.15%)



Average Accuracy:

- Caltech101+Caltech256+Cifar10/3 =
 $(93.23+85.14+94.16)/3 \approx \underline{\underline{91}}$

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.