Deep Learning & Ensemble Learning

Instructor – Dr. Yimin Yang

Presented by –
Chirag Patel (0883437)
Dhruvi Desai (0884718)
Priyank Shah (0883445)
Vatsal Patel (0883974)

Description:

In this course project we have focused on the following two aspects.

1) Implement the basic deep convolutional neural networks to an advanced GPU-based environment- Matlab

2) Classification methods deep convolutional neural networks and Extreme Learning Machine are used to obtain the average top-1 accuracy

3) Any feature extraction methods are allowed to use to boost the performance such as PCA, autoencoder, deep believe networks, etc.

Dataset: CIFAR10 – https://www.cs.toronto.edu/~kriz/cifar.html

This dataset consist of 10 classes with 60000 thousand images per class. This data is divided into 10000 testing data and 50000 training data which further is divided five training batches in random order. The classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck which are completely mutually exclusive.

Before we begin the code and execute the train and test models, we need to install the following packages into Matlab:
1. Deep learning Toolbox
2. Parallel Toolbox
3. VGG16 (For pre-trained model)
4. Resnet (For pre-trained model)
5. Static Machine Learning Toolbox

Pre-trained model execution:

We will use the pre-trained model VGG 16 and Resnet to train and test the dataset "CIFAR10" in the first phase. We have used Matlab R2018a for programming in this project. Files needed for the execution:

1) downloadCIFAR10.m – This file is used to download the images from url and extract the data as the data from Cifar10 is not in the form of 2-D images.
2) deepfeature_cifar10.m – This file is used to execute the training and testing on the network and obtain deep features on the dataset CIFAR10 and then it is feedforwarded to classifier ELM.
3) readAndPreprocessImage.m– This file is used to convert the image size to the required network to provide the right size of the network.
4) ELM.m -Extreme learning machine is a shallow classifier, which is a new learning algorithm for the single hidden layer feedforward neural networks.

How Extreme Learning Machine works as softmax classifier?

ELMs involve randomly assigning weights in the hidden units and train extremely fast compared to traditional neural networks. A hidden layer of 100 units was used in our experiments. We have used kernel Extreme Learning Machines (ELM) due to the learning speed and accuracy of the algorithm. ELM proposes a simple and robust learning algorithm for single-hidden layer feedforward networks. The input layer's bias and weights are initialized randomly to obtain the output of the second (hidden) layer. The bias and weights of the second layer are calculated by a simple generalized inverse operation of the hidden layer output matrix

Working:

In this project, we propose a hybrid model consisting of a Deep Convolutional feature extractor followed by a fast and accurate classifier, the Extreme Learning Machine, for detecting the images. Deep CNNs used for image classification take a very long time to train that is why we have used this model. Also with pre-trained models, the fully connected layers need to be trained with backpropagation, which can be very slow. We have used Extreme Learning Machine (ELM) in project as the final classifier trained on pre-trained Deep CNN feature extractor. We use state of the art Deep CNNs: VGG16 and Resnet101 and replace the softmax classifier with the ELM classifier. For both the VGG16 and Resnet101, the number of fully connected layers is also reduced. Especially in VGG16, which has 3 fully connected layers of 4096 neurons each followed by a softmax classifier, we replace two of these with an ELM classifier. The difference is convergence rate between fine-tuning the fully connected layers of pre-trained models. Extracting features from two DCNN networks help us to boost and provide accurate training and testing of data which lead us to generate a higher accuracy.

Code Workflow:

First, we download the dataset using the downloadCIFAR10.m file and store it in the path provided below.

```
outputFolder = fullfile('C:\Users\vpatel25\Desktop\CIFAR10');

trainDigitFolder = fullfile(outputFolder, 'cifar10Train');
testDigitFolder =  fullfile(outputFolder, 'cifar10Test');
trainDigitData =
imageDatastore(trainDigitFolder,'IncludeSubfolders',true,'LabelSource','folde
rnames');
testDigitData =
imageDatastore(testDigitFolder,'IncludeSubfolders',true,'LabelSource','folder
names');
```

Deep feature extraction from VGG16:

```
%Deep Feature extraction
```

```
vgg_trainingFeatures = activations(net, trainDigitData,
'drop7','MiniBatchSize',100);
vgg_trainingFeatures=reshape(vgg_trainingFeatures,[1*1*4096,size(vgg_training
Features,4)])';
vgg_testingFeatures = activations(net,
testDigitData,'drop7','MiniBatchSize',100);
vgg_testingFeatures=reshape(vgg_testingFeatures,[1*1*4096,size(vgg_testingFea
tures,4)])';
```

Deep feature extraction from Resnet101:

```
resnet_trainingFeatures = activations(net, trainDigitData,
'pool5','MiniBatchSize',100);
resnet_trainingFeatures=reshape(resnet_trainingFeatures,[1*1*2048,size(resnet
_trainingFeatures,4)])';
resnet_testingFeatures = activations(net,
testDigitData,'pool5','MiniBatchSize',100);
resnet_testingFeatures=reshape(resnet_testingFeatures,[1*1*2048,size(resnet_t
estingFeatures,4)])';
```

Feature Concatenation:

```
new_F_train = horzcat(vgg_trainingFeatures,resnet_trainingFeatures);
new_F_test = horzcat(vgg_testingFeatures, resnet_testingFeatures);
```

Label the concatenated data:

```
Training=[train_label new_F_train];  % tr_label represents training label,
tr_fea represents training features
Testing=[test_label new_F_test];  % the same as above
```

Forwarding the feature to ELM classifier:

```
[train_time,
train_accuracy,test_accuracy1]=ELM(Training,Testing,1,1000,'sig',2^2);
```

Output 1:

Editor - C:\Users\vpatel25\Desktop\deepfeature_cifar100.m

Command Window

```
  CUDADevice with properties:

                      Name: 'GeForce GTX 1060 6GB'
                     Index: 1
         ComputeCapability: '6.1'
            SupportsDouble: 1
             DriverVersion: 10.2000
            ToolkitVersion: 9
        MaxThreadsPerBlock: 1024
          MaxShmemPerBlock: 49152
        MaxThreadBlockSize: [1024 1024 64]
               MaxGridSize: [2.1475e+09 65535 65535]
                 SIMDWidth: 32
               TotalMemory: 6.4425e+09
           AvailableMemory: 5.2519e+09
       MultiprocessorCount: 10
              ClockRateKHz: 1708500
               ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1


TrainingTime =

    5.7031


TestingTime =

    0.7500


TrainingAccuracy =

    0.9043


TestingAccuracy =

    0.8934

>>
```

Training Accuracy: 90.43

Testing Accuracy: 89.34

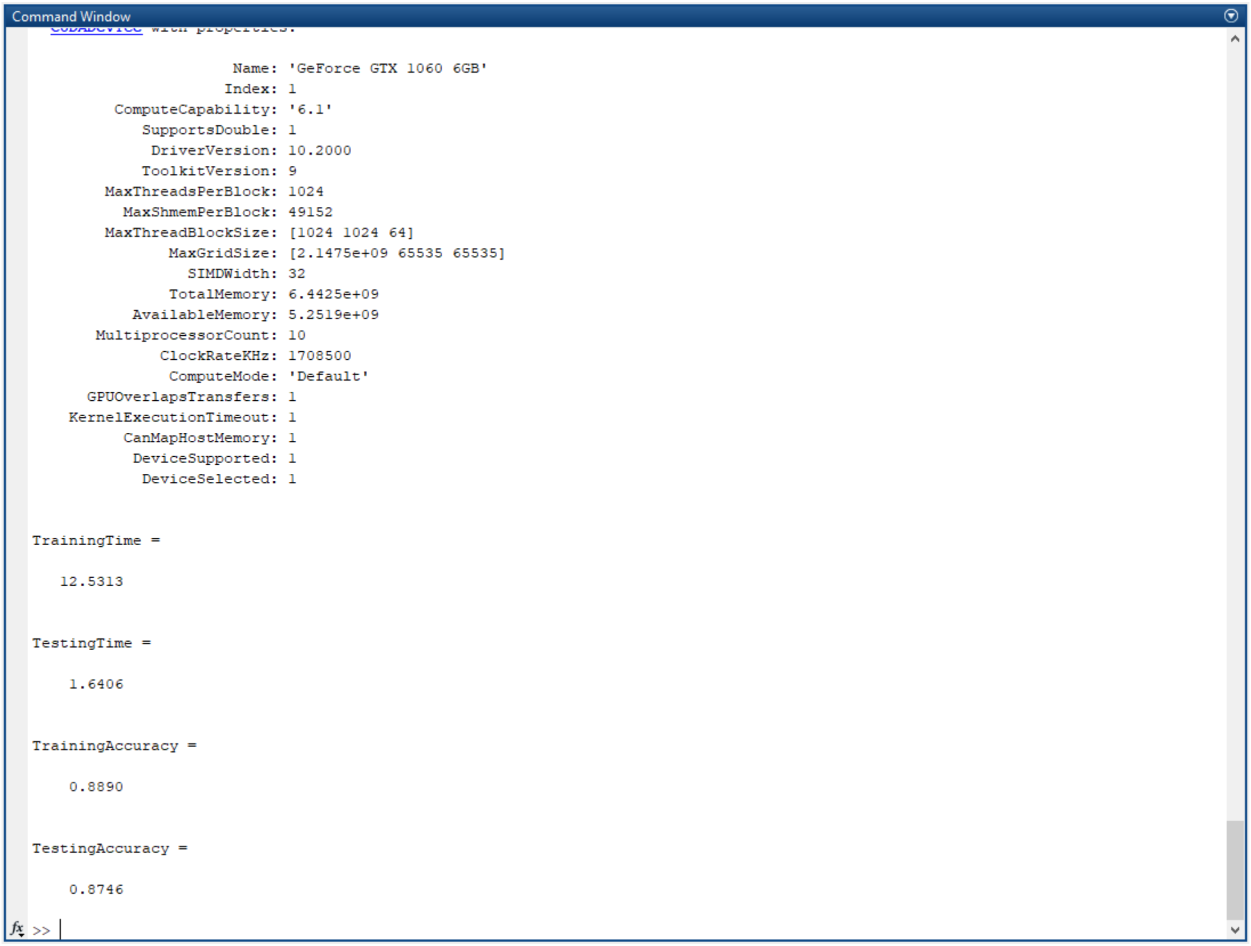
<u>Output 2:</u>

```
Editor - C:\Users\vpatel25\Desktop\cifar100.m
Command Window
  CUDADevice with properties:

                      Name: 'GeForce GTX 1060 6GB'
                     Index: 1
         ComputeCapability: '6.1'
            SupportsDouble: 1
             DriverVersion: 10.2000
            ToolkitVersion: 9
        MaxThreadsPerBlock: 1024
          MaxShmemPerBlock: 49152
        MaxThreadBlockSize: [1024 1024 64]
               MaxGridSize: [2.1475e+09 65535 65535]
                 SIMDWidth: 32
               TotalMemory: 6.4425e+09
           AvailableMemory: 5.2519e+09
       MultiprocessorCount: 10
              ClockRateKHz: 1708500
               ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1


TrainingTime =

   12.6250


TestingTime =

    1.6875


TrainingAccuracy =

    0.8856


TestingAccuracy =

    0.8767
```

Training Accuracy: 88.56

Testing Accuracy: 87.67

Output 3:



```
                   Name: 'GeForce GTX 1060 6GB'
                  Index: 1
      ComputeCapability: '6.1'
         SupportsDouble: 1
          DriverVersion: 10.2000
         ToolkitVersion: 9
     MaxThreadsPerBlock: 1024
       MaxShmemPerBlock: 49152
     MaxThreadBlockSize: [1024 1024 64]
            MaxGridSize: [2.1475e+09 65535 65535]
              SIMDWidth: 32
            TotalMemory: 6.4425e+09
        AvailableMemory: 5.2519e+09
    MultiprocessorCount: 10
           ClockRateKHz: 1708500
            ComputeMode: 'Default'
   GPUOverlapsTransfers: 1
 KernelExecutionTimeout: 1
       CanMapHostMemory: 1
        DeviceSupported: 1
         DeviceSelected: 1


TrainingTime =

   12.5313


TestingTime =

    1.6406


TrainingAccuracy =

    0.8890


TestingAccuracy =

    0.8746

>>
```

Training Accuracy: 88.90

Testing Accuracy: 87.46

Average of 3 runs:

Average Training Accuracy: 89.29

Average Testing Accuracy: 88.15

Conclusion:

When we tried classifying using only one DCNN feature extracting i.e., VGG16 but the testing accuracy reached around 83%, whereas when compared to two feature extracting we get a higher accuracy.