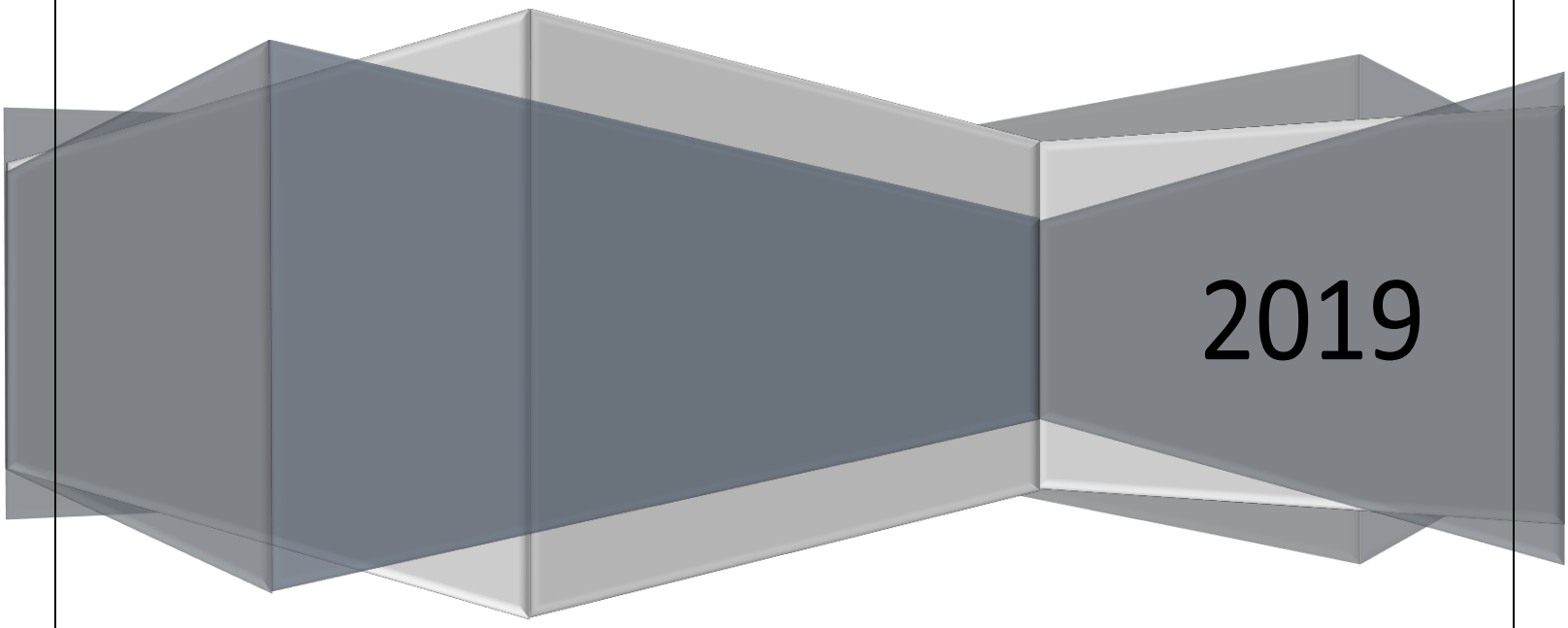
Lakehead UNIVERSITY

CS 5413-WE: Neural Networks and Machine learning

Hardil Mehta (0898141)



CONTENTS

SPECIFICATIONS:

Language used: Matlab

Operating System: Windows 10

Used Dataset:

1. Caltech101
2. Caltech256
3. Cifar10
4. Cifar100

GPU:

- NVIDIA GTX 765M (2 GB)
- NVIDIA GTX 1060 (6 GB)

PROJECT IMPLEMENTATION

DATASET 1: CALTECH101

CODE

```
%% Download the dataset

% Location of the compressed data set
url = 'http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_ObjectCat-
egories.tar.gz';

% Store the output in a temporary folder
outputFolder = fullfile('E:\Lakehead\SEM 2\Neural networks\Project\da-
taset\caltech', 'caltech101'); % define output folder
if ~exist(outputFolder, 'dir') % download only once
    disp('Downloading 126MB Caltech101 data set...');
    untar(url, outputFolder);
end

disp('12 steps to output')
%% Create ImageDatastore of the dataset for processing in Matlab.
rootFolder = fullfile(outputFolder, '101_ObjectCategories');
imds = imageDatastore(fullfile(rootFolder), 'LabelSource', 'foldernames','In-
cludeSubfolders',true);
clear url outputFolder rootFolder;

%% Split each label
% Using 30 images for training and rest for testing
[trainingSet, validationSet] = splitEachLabel(imds, 30);
disp('1. preprocessing training image for resnet101');
trainingSet.ReadFcn = @(filename)readAndPreprocessImageForGoogle(filename);
%redefine read function to process images while read
disp('2. preprocessing testing image for resnet101');
validationSet.ReadFcn = @(filename)readAndPreprocessImageForGoogle(filename);
%redefine read function to process images while read
clear imds
%% Load resnet
disp('3. Loading Pretrained Resnet');
 net = resnet101;
 %% Get features from resnet
 disp('4. Loading Resnet train features');
 gpuDevice(1)
 resnet_features_train = activations(net,trainingSet,'fc1000','MiniBatch-
Size',120);
 disp('5. Loading Resnet test features');
 resnet_features_test = activations(net,validationSet,'fc1000','MiniBatch-
Size',120);
 resnet_features_train = reshape(resnet_features_train,[1*1*1000,size(res-
net_features_train,4)])' ;
 resnet_features_test = reshape(resnet_features_test,[1*1*1000,size(res-
net_features_test,4)])';
```

Code 1: Code for loading dataset, and getting features from resnet101.

```
%% Load inceptionv3
disp('6. preprocessing training image for inceptionv3');
trainingSet.ReadFcn = @(filename)readAndPreprocessImage(filename); %redefine
read function to process images while read
disp('7. preprocessing testing image for inceptionv3');
validationSet.ReadFcn = @(filename)readAndPreprocessImage(filename);
%redefine read function to process images while read
net = inceptionv3;


%% Get training set deep features from inceptionv3
gpuDevice(1)
disp('8. Loading inceptionv3 train features');
inceptionv3_features_train =
activations(net,trainingSet,'avg_pool','MiniBatchSize',120);
%% Get inceptionv3 test deep Features
disp('9. Loading inceptionv3 test features');
gpuDevice(1);
clear ans
inceptionv3_features_test =
activations(net,validationSet,'avg_pool','MiniBatchSize',120);
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)
])';

%% Merge Resnet and inceptionv3 deep features for training and testing
disp('10. Combining the features from inceptionv3 and resnet');
new_F_train = horzcat(inceptionv3_features_train, resnet_features_train);
new_F_test = horzcat(inceptionv3_features_test, resnet_features_test);

clear inceptionv3_features_train resnet_features_train
inceptionv3_features_test resnet_features_test net;
%%
train_labels = grp2idx(trainingSet.Labels);
test_labels = grp2idx(validationSet.Labels);
%%
disp('11. creating training and testing dataset for elm');
 training = horzcat(train_labels,new_F_train);
 testing = horzcat( test_labels,new_F_test);
 clear train_labels new_F_train test_labels new_F_test;
%%
C = 2^-12;
disp('12. Classification using ELM');
[TrainingTime, TestingAccuracy,Training,Testing] = ELM(training, testing, 1,
10000, 'sig',C);
```

Code 1 (continue): Code for getting features from inception v3 and using deep features in ELM classifier.

STRUCTURE USED:

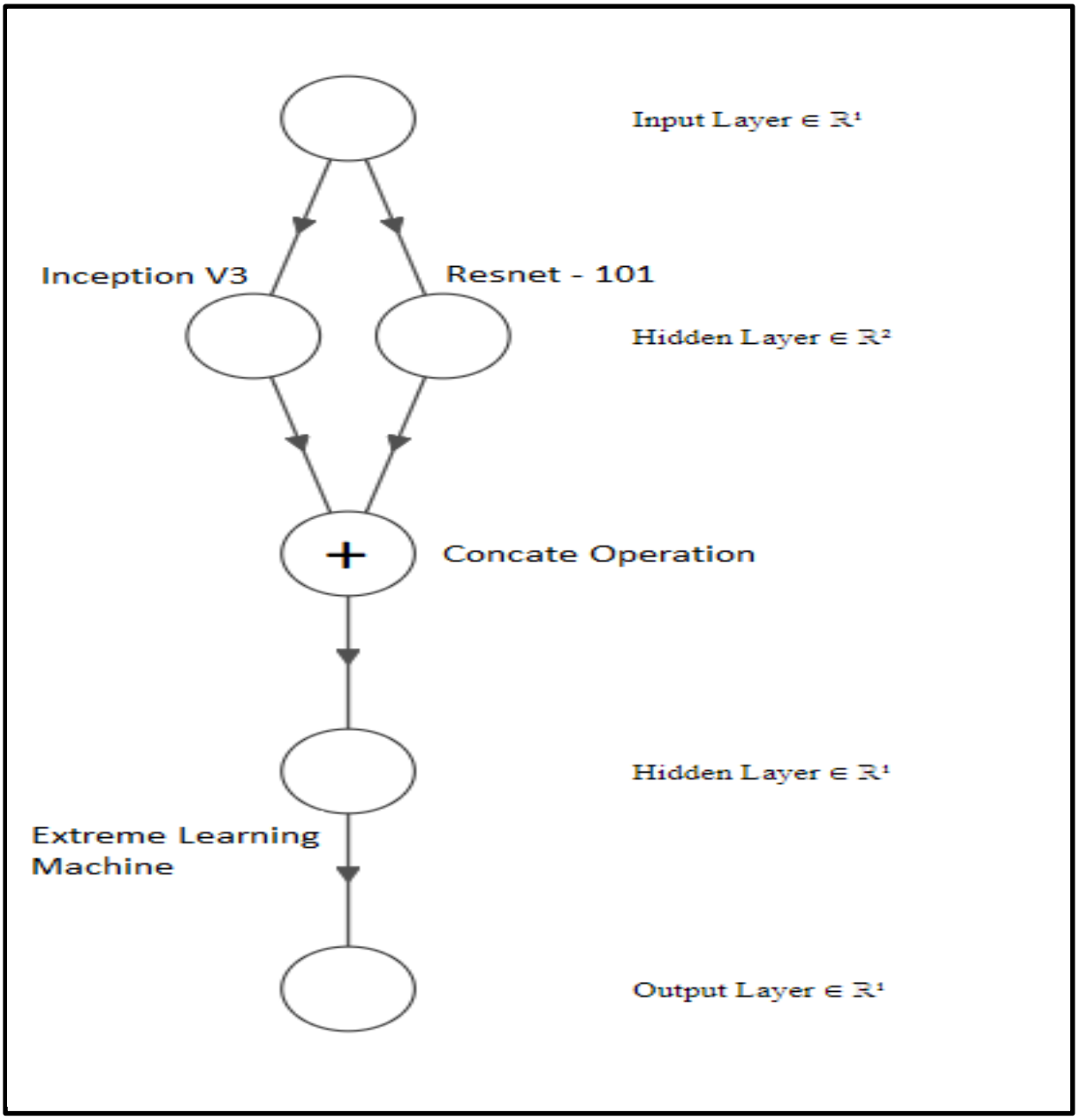


Figure 1. Structure for Classification on Caltech 101.

The Following structure is used for classification. Deep features are obtained from Inception V3 and Resnet 101. These Deep features are then combined. Extreme Learning Machine (ELM) is used as a classifier and is trained on this data and is used for classification. The Output of the class is provided by the ELM.

RESULTS

- **Average top-1 Accuracy: 89.66 %**
- **Accuracy in each Trials: 89.53 %, 89.68% and 89.78%**
- **Training Speed: ~ 1.5 hours (NVIDIA GTX 765M)**

OUTPUT:

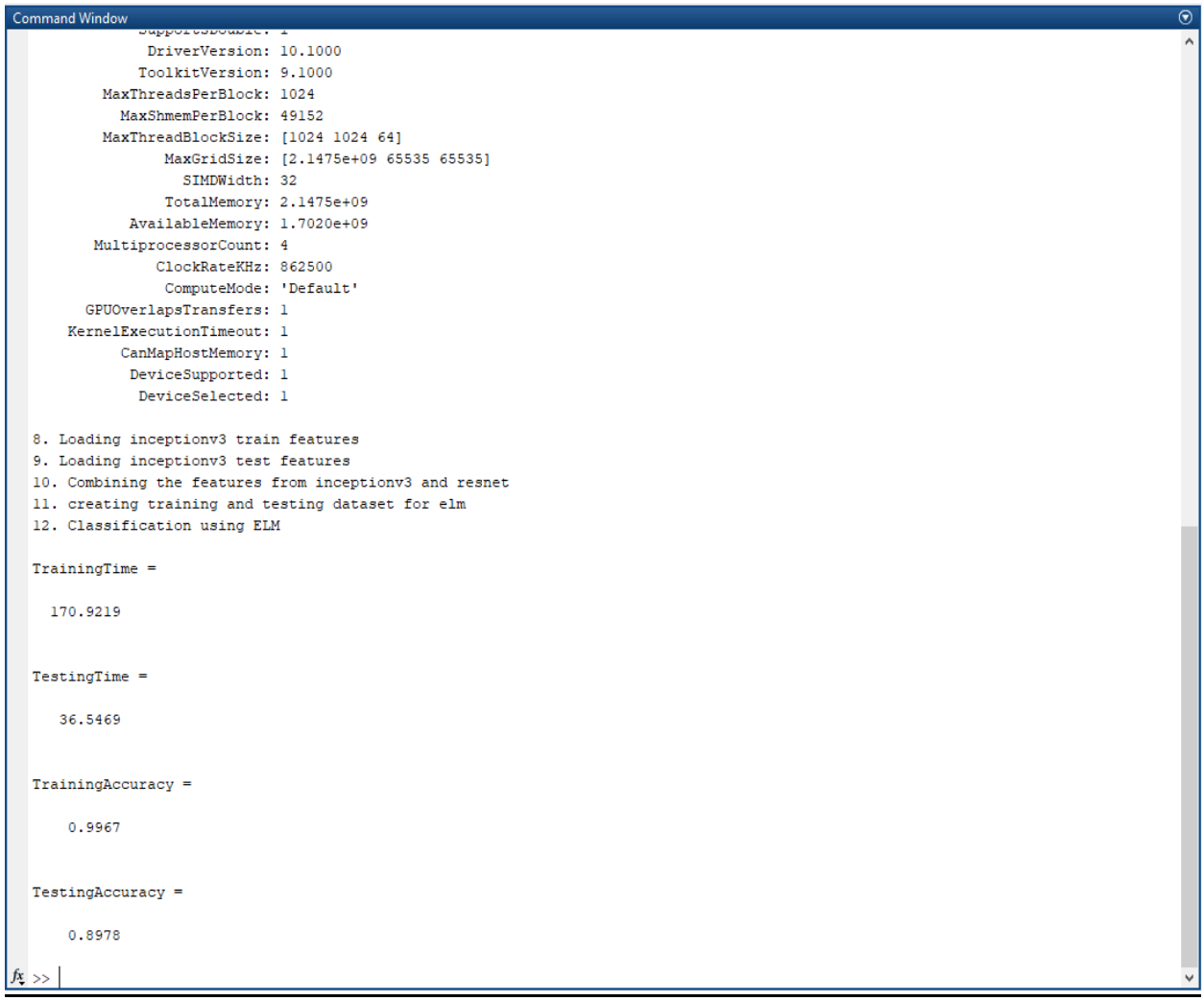


Figure 2. OUTPUT on Caltech 101.

CALTECH256:

CODE:

```
 %% Download the dataset

% Location of the compressed data set
url = 'E:\Lakehead\SEM 2\Neural networks\Project\256_ObjectCategories.tar';

% Store the output in a temporary folder
outputFolder = fullfile('E:\Lakehead\SEM 2\Neural networks\Project\cal-
tech256'); % define output folder

if ~exist(outputFolder, 'dir')
    disp('Extracting Caltech256 data set...');
    untar(url, outputFolder);
end

disp('12 steps to output')
%% create imagedatastore
imagefolders = fullfile(outputFolder, '256_ObjectCategories');
imds = imageDatastore(fullfile(imagefolders), 'LabelSource', 'folder-
names','IncludeSubfolders',true);
clear url outputFolder imagefolders;
%% Split each label
% Using 30 images for training and rest for testing
[trainingSet, testingSet] = splitEachLabel(imds, 30);
disp('1. preprocessing training image for resnet101');
trainingSet.ReadFcn = @(filename)readAndPreprocessImageForGoogle(filename);
%redefine read function to process images while read
disp('2. preprocessing testing image for resnet101');
testingSet.ReadFcn = @(filename)readAndPreprocessImageForGoogle(filename);
%redefine read function to process images while read
clear imds
%% Load resnet
disp('3. Loading Pretrained Resnet');
 net = resnet101;
 %% Get features from resnet
 disp('4. Loading Resnet train features');
 gpuDevice(1)
 resnet_features_train = activations(net,trainingSet,'fc1000','MiniBatch-
Size',50);
 disp('Loading Resnet test features');
 disp('5. Loading Resnet test features');
 resnet_features_test = activations(net,testingSet,'fc1000','MiniBatch-
Size',50);
 resnet_features_train = reshape(resnet_features_train,[1*1*1000,size(res-
net_features_train,4)])' ;
 resnet_features_test = reshape(resnet_features_test,[1*1*1000,size(res-
net_features_test,4)])';
```

Code 2: Code for loading dataset, and getting features from resnet101.

```
%% Load inceptionv3
disp('6. preprocessing training image for inceptionv3');
trainingSet.ReadFcn = @(filename)readAndPreprocessImage(filename); %redefine
read function to process images while read
disp('7. preprocessing testing image for inceptionv3');
validationSet.ReadFcn = @(filename)readAndPreprocessImage(filename);
%redefine read function to process images while read
net = inceptionv3;


%% Get training set deep features from inceptionv3
gpuDevice(1)
disp('8. Loading inceptionv3 train features');
inceptionv3_features_train =
activations(net,trainingSet,'avg_pool','MiniBatchSize',120);
%% Get inceptionv3 test deep Features
disp('9. Loading inceptionv3 test features');
gpuDevice(1);
clear ans
inceptionv3_features_test =
activations(net,validationSet,'avg_pool','MiniBatchSize',120);
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)
])';

%% Merge Resnet and inceptionv3 deep features for training and testing
disp('10. Combining the features from inceptionv3 and resnet');
new_F_train = horzcat(inceptionv3_features_train, resnet_features_train);
new_F_test = horzcat(inceptionv3_features_test, resnet_features_test);

clear inceptionv3_features_train resnet_features_train
inceptionv3_features_test resnet_features_test net;
%%
train_labels = grp2idx(trainingSet.Labels);
test_labels = grp2idx(validationSet.Labels);
%%
disp('11. creating training and testing dataset for elm');
 training = horzcat(train_labels,new_F_train);
 testing = horzcat( test_labels,new_F_test);
 clear train_labels new_F_train test_labels new_F_test;
%%
C = 2^-12;
disp('12. Classification using ELM');
[TrainingTime, TestingAccuracy,Training,Testing] = ELM(training, testing, 1,
10000, 'sig',C);
```

Code 2(continue): Code for getting features from inception v3 and using deep features in ELM classifier.

STRUCTURE USED:

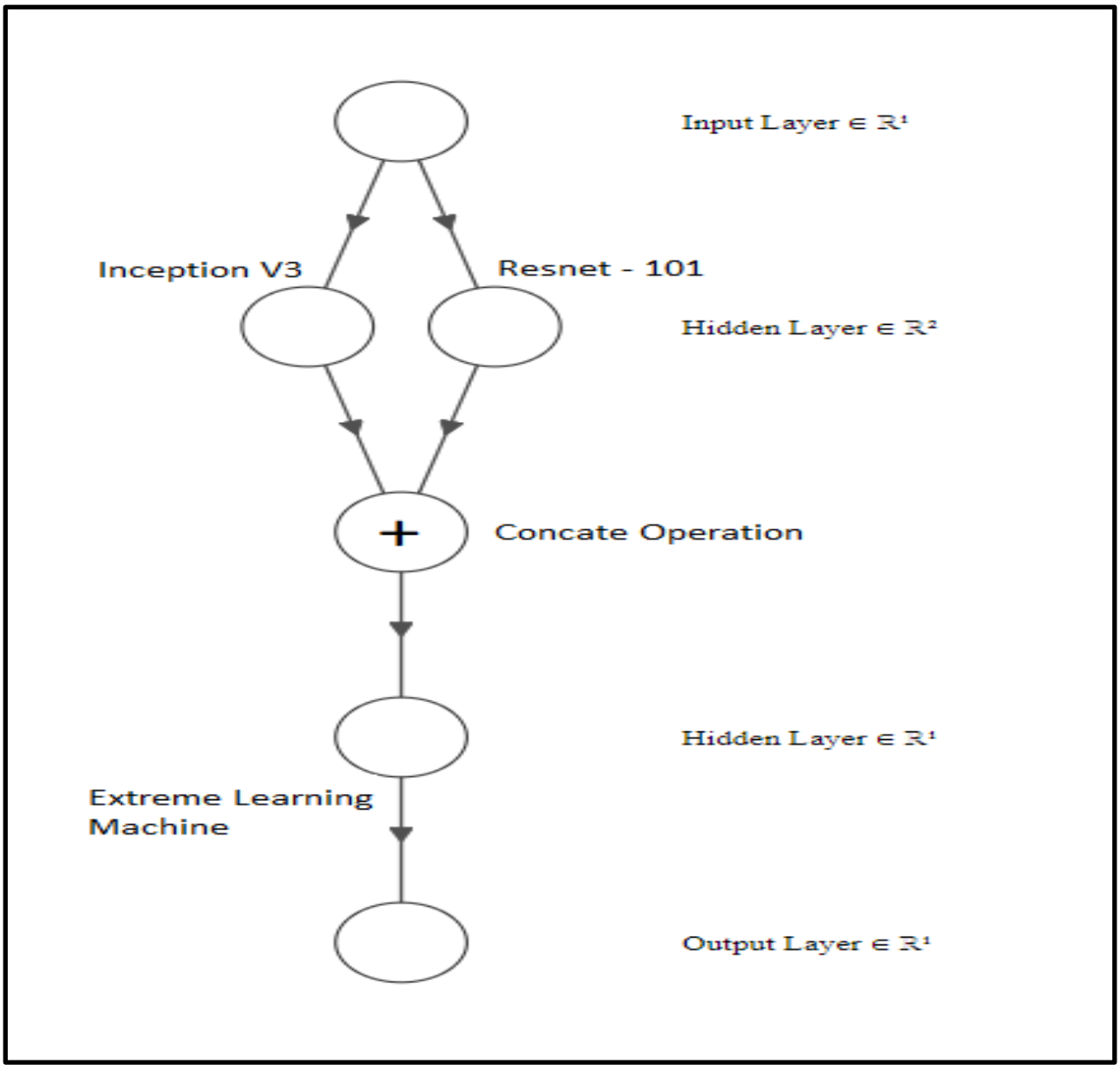


Figure 1. Structure for Classification on Caltech 256.

The structure used is same from Caltech101.

The Following structure is used for classification. Deep features are obtained from Inception V3 and Resnet 101. These Deep features are then combined. Extreme Learning Machine (ELM) is used as a classifier and is trained on this data and is used for classification. The Output of the class is provided by the ELM.

RESULTS

- **Average top-1 Accuracy: 80.37 %**
- **Accuracy in each Trials: 80.74 %, 80.26% and 80.12%**
- **Training Speed: ~ 4 hours (NVIDIA GTX 765M)**

OUTPUT:

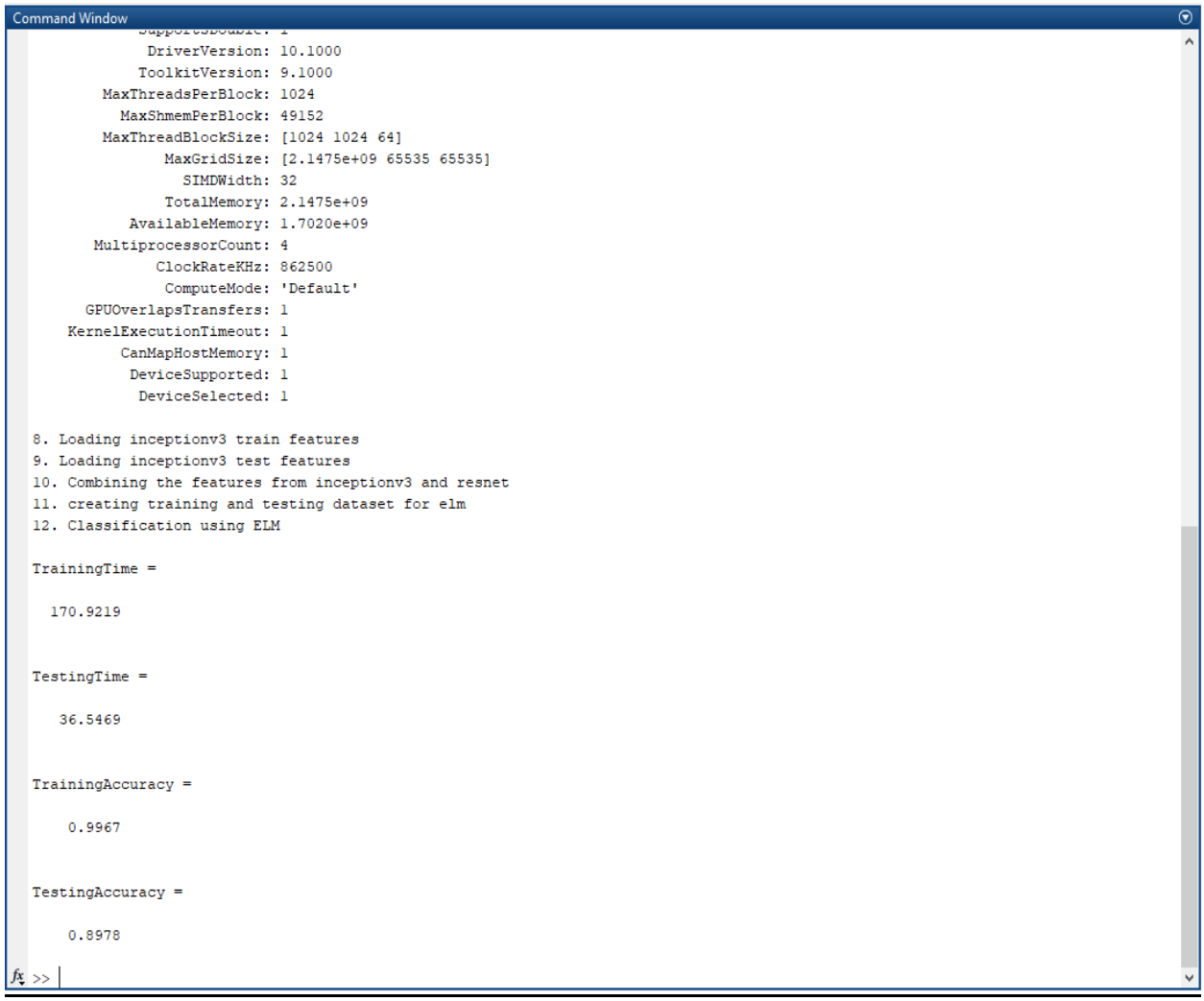


Figure 2. OUTPUT on Caltech 256.

CIFAR10:

CODE:

```
%%

% Enter the location of Dataset
outputFolder = fullfile('C:\Users\Student\Desktop\Neural\Cifar10'); % define
output folder


%% Load DataSet
trainFolder = fullfile(outputFolder, 'cifar10Train');
testFolder =  fullfile(outputFolder, 'cifar10Test');
trainingSet = imageDatastore(fullfile(trainFolder), 'LabelSource', 'folder-
names','IncludeSubfolders',true);
validationSet = imageDatastore(fullfile(testFolder), 'LabelSource', 'folder-
names','IncludeSubfolders',true);

%% Preprocess Images for GoogLeNet
trainingSet.ReadFcn = @(filename)readAndPreprocessImageForGoogle(filename);
%redefine read function to process images while read
validationSet.ReadFcn = @(filename)readAndPreprocessImageForGoogle(filename);
%redefine read function to process images while read
%% Transfer the last three layers for learning.
lrate=20;
miniBatchSize = 120;
net = googlenet;
lgraph = layerGraph(net);
lgraph = removeLayers(lgraph, {'loss3-classifier','prob','output'});%discard
output layers
numClasses = numel(categories(trainingSet.Labels));%Set the fully connected
layer to the same size as the number of classes in the new data sat.
newLayers = [
    fullyConnectedLayer(numClasses,'Name','fc','WeightLearnRateFac-
tor',lrate,'BiasLearnRateFactor', lrate)%set the learning rate of new layers
    softmaxLayer('Name','softmax')
    classificationLayer('Name','classoutput')];

lgraph = addLayers(lgraph,newLayers);
lgraph = connectLayers(lgraph,'pool5-drop_7x7_s1','fc'); %add the new output
layers to the pretrained CNN
figure('Units','normalized','Position',[0.3 0.3 0.4 0.4]);
plot(lgraph)
ylim([0,10])
```

Code 3: Code for loading dataset and changing the layers for transfer Learning on GoogLeNet.

```
%%
%training options
options = trainingOptions('sgdm',...
    'MiniBatchSize',miniBatchSize,... %set mini batch size
      'LearnRateSchedule','piecewise',...
      'LearnRateDropFactor',0.1,...
      'LearnRateDropPeriod',2,...
      'MaxEpochs',6,...
      'InitialLearnRate',1e-3,...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','auto');
%% Train the network using the training data.

net = trainNetwork(trainingSet,lgraph,options);
save GoogLeNetCifar10 net
clear lgraph lrate miniBatchSize newLayers numClasses outputFolder testFolder
trainFolder options;
%% Predict Output
predictedLabels = classify(net,validationSet);
fin_accuracy = mean(predictedLabels == validationSet.Labels);
fprintf('Accuracy of ImageNet Pretrained GoogLeNet: %s \n', fin_accuracy);
```

Code 3 (continue): Code for training Options and classification using GoogLeNet.

STRUCTURE USED:

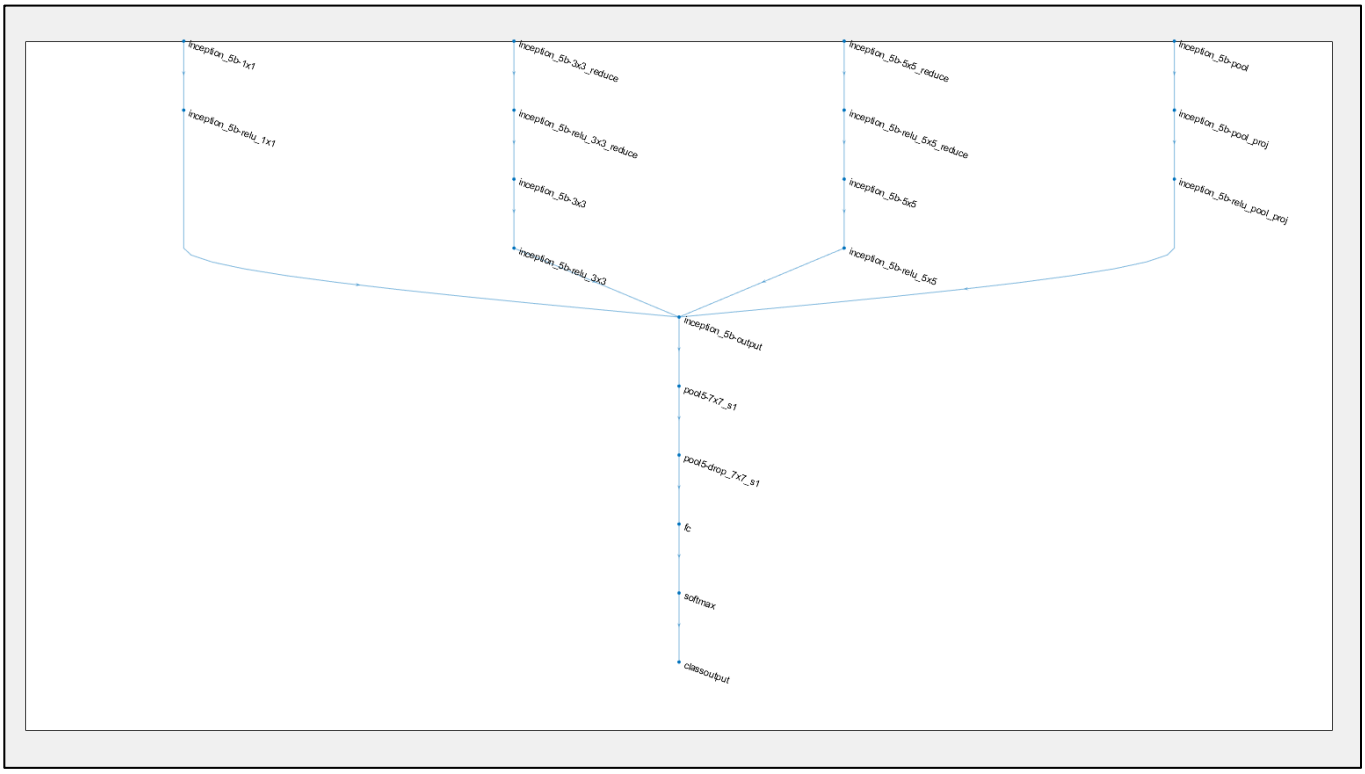


Fig 3. Structure of GoogLeNet with new Fully-connected layer, softmax Layers and cross-entropy layer

Here, Transfer Learning is used on Cifar10 using GoogLeNet. We change the last three layers to learn the new dataset and use the weights from ImageNet Pretrained network and train the network on new dataset.

RESULTS:

- **Average top-1 Accuracy: 94.31 %**
- **Accuracy in each Trials: 94.53 %, 94.28% and 94.12%**
- **Training Speed: ~ 1.25 hours (NVIDIA GTX 1060)**

OUTPUT:

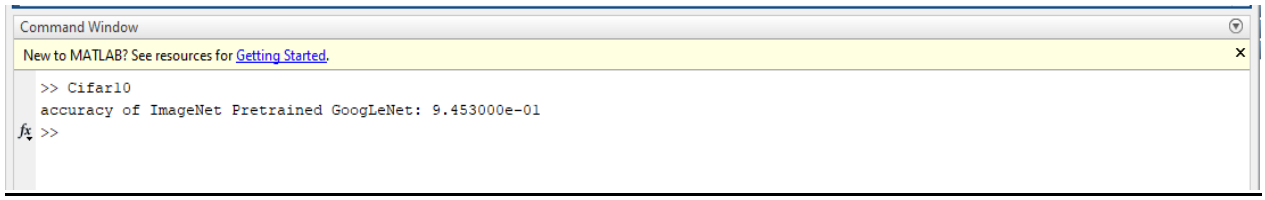


Figure 2. OUTPUT on Cifar10.

CIFAR100:

CODE:

```
%% Transfer Learning on Cifar100 using GoogLeNet.

% Select the location of the dataset
outputFolder = fullfile('E:\Lakehead\SEM 2\Neural networks\Pro-
ject\Cifar100\cifar-100-matlab\CIFAR-100'); % define output folder


%%
trainFolder = fullfile(outputFolder, 'TRAIN');
testFolder =  fullfile(outputFolder, 'TEST');
trainingSet = imageDatastore(fullfile(trainFolder), 'LabelSource', 'folder-
names','IncludeSubfolders',true);
TestSet = imageDatastore(fullfile(testFolder), 'LabelSource', 'folder-
names','IncludeSubfolders',true);

%%
trainingSet.ReadFcn = @(filename)readAndPreprocessImageForGoogle(filename);
%redefine read function to process images while read
TestSet.ReadFcn = @(filename)readAndPreprocessImageForGoogle(filename); %re-
define read function to process images while read
%%
lrate=20;
miniBatchSize = 30;
net = googlenet;
lgraph = layerGraph(net);
lgraph = removeLayers(lgraph, {'loss3-classifier','prob','output'});%discard
output layers
numClasses = numel(categories(trainingSet.Labels));%Set the fully connected
layer to the same size as the number of classes in the new data sat.
newLayers = [
    fullyConnectedLayer(numClasses,'Name','fc','WeightLearnRateFac-
tor',lrate,'BiasLearnRateFactor', lrate)%set the learning rate of new layers
    softmaxLayer('Name','softmax')
    classificationLayer('Name','classoutput')];

lgraph = addLayers(lgraph,newLayers);
lgraph = connectLayers(lgraph,'pool5-drop_7x7_s1','fc'); %add the new output
layers to the pretrained CNN
figure('Units','normalized','Position',[0.3 0.3 0.4 0.4]);
plot(lgraph)
ylim([0,10])
```

Code 4: Code for loading dataset and changing the layers for transfer Learning on GoogLeNet.

```
%%
%training options
options = trainingOptions('sgdm',...
    'MiniBatchSize',miniBatchSize,... %set mini batch size
      'LearnRateSchedule','piecewise',...
      'LearnRateDropFactor',0.1,...
      'LearnRateDropPeriod',3,...
      'MaxEpochs',12,...
      'InitialLearnRate',1e-3,...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','auto');
%% Train the network using the training data.
ans = gpuDevice(1);
clear ans;
net = trainNetwork(trainingSet,lgraph,options);
save GoogLeNetCifar100 net
clear lgraph lrate miniBatchSize newLayers numClasses outputFolder testFolder
trainFolder options
%%
predictedLabels = classify(net,TestSet);
fin_accuracy = mean(predictedLabels == TestSet.Labels);
fprintf('accuracy of ImageNet Pretrained GoogLeNet: %s \n', fin_accuracy);
```

Code 3 (continue): Code for training Options and classification using GoogLeNet.

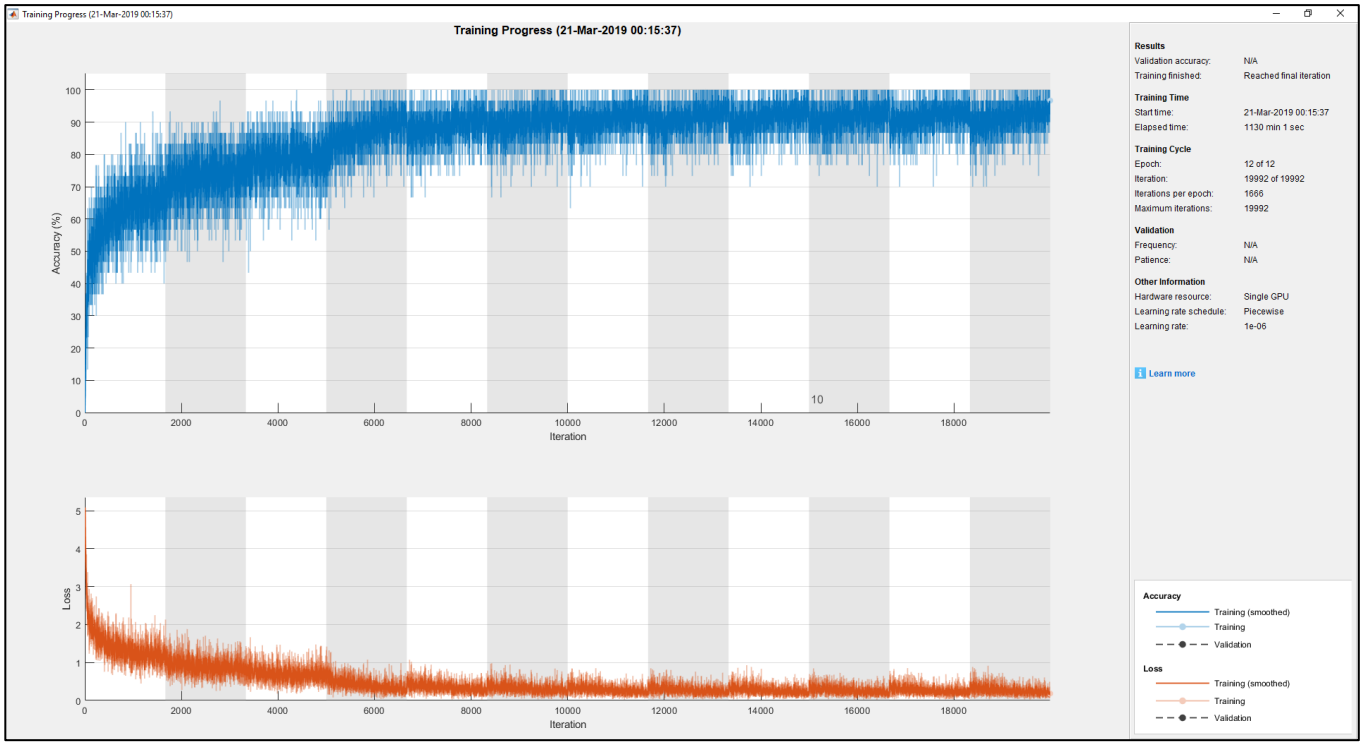


Fig 4. Training process on GoogLeNet.

STRUCTURE USED:

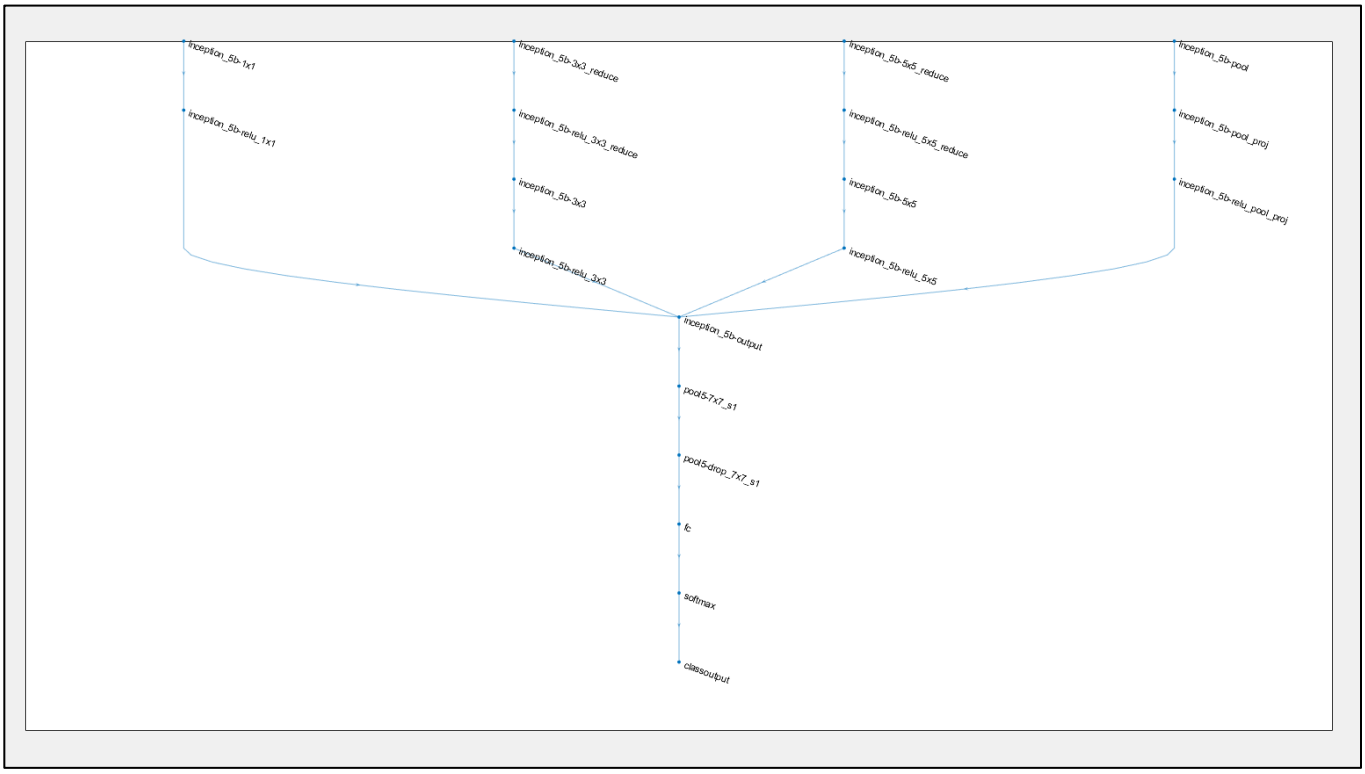


Fig 4. Structure of GoogLeNet with new Fully-connected layer, softmax Layers and cross-entropy layer

Here, Transfer Learning is used on Cifar100 using GoogLeNet. We change the last three layers to learn the new dataset and use the weights from ImageNet Pretrained network and train the network on new dataset.

RESULTS:

- **Average top-1 Accuracy: 79.51 %**
- **Accuracy in each Trials: 79.54 %, 79.36% and 79.66%**
- **Training Speed: ~ 19.5 hours (NVIDIA GTX 765M)**

OUTPUT:

```
>> Cifar100
30            end
accuracy of ImageNet Pretrained GoogLeNet: 7.954000e-01
>>
```

Figure 2. OUTPUT on Cifar100.