i)

```
net=vgg16;
```

ii)

```
Arch=cell(numel(net.Layers),5);
for i=[2 4 7 9 12 14 16 19 21 23 26 28 30]
    l=net.Layers(i);
    Arch{i,1}=l.Name;
    Arch{i,2}='Convolution2D';
    Arch{i,3}=l.NumFilters;
    Arch{i,4}=l.FilterSize;
    Arch{i,5}=numel(l.Weights)+numel(l.Bias);
end
for i=[3 5 8 10 13 15 17 20 22 24 27 29 31 34 37]
    l=net.Layers(i);
    Arch{i,1}=l.Name;
    Arch{i,2}='ReLU';
end
for i=[6 11 18 25 32]
    l=net.Layers(i);
    Arch{i,1}=l.Name;
    Arch{i,2}='MaxPooling';
end
for i=[33 36 39]
    l=net.Layers(i);
    Arch{i,1}=l.Name;
    Arch{i,2}='FullyConnected';
    Arch{i,5}=numel(l.Weights)+numel(l.Bias);
end
for i=[35 38]
    l=net.Layers(i);
    Arch{i,1}=l.Name;
    Arch{i,2}='DropoutLayer';
end
l=net.Layers(40);
Arch{40,1}=l.Name;
Arch{40,2}='Softmax';
l=net.Layers(41);
Arch{41,1}=l.Name;
Arch{41,2}='ClassificationOutput';
l=net.Layers(1);
Arch{1,1}=l.Name;
Arch{1,2}='ImageInput';
table(Arch)
```

ans = 41×1 table

	Arch				
1	'input'	'ImageInput'	[]	[]	[]
2	'conv1_1'	'Convoluti...	64	[3,3]	1792
3	'relu1_1'	'ReLU'	[]	[]	[]
4	'conv1_2'	'Convoluti...	64	[3,3]	36928
5	'relu1_2'	'ReLU'	[]	[]	[]

6	'pool1'	'MaxPooling'	[]	[]	[]
7	'conv2_1'	'Convoluti...	128	[3,3]	73856
8	'relu2_1'	'ReLU'	[]	[]	[]
9	'conv2_2'	'Convoluti...	128	[3,3]	147584
10	'relu2_2'	'ReLU'	[]	[]	[]

This network has 41 layers which is more than twice the number of layers in MNIST. Both networks have input layer, convolutionl layers, relu layers, maxpooling layers, classification layer, softmax and fully connected layers. MNIST has batch layers and VGG16 has dropout layers exclusively. MNIST has 54666 parameters and VGG16 has more than 138 million parameters.

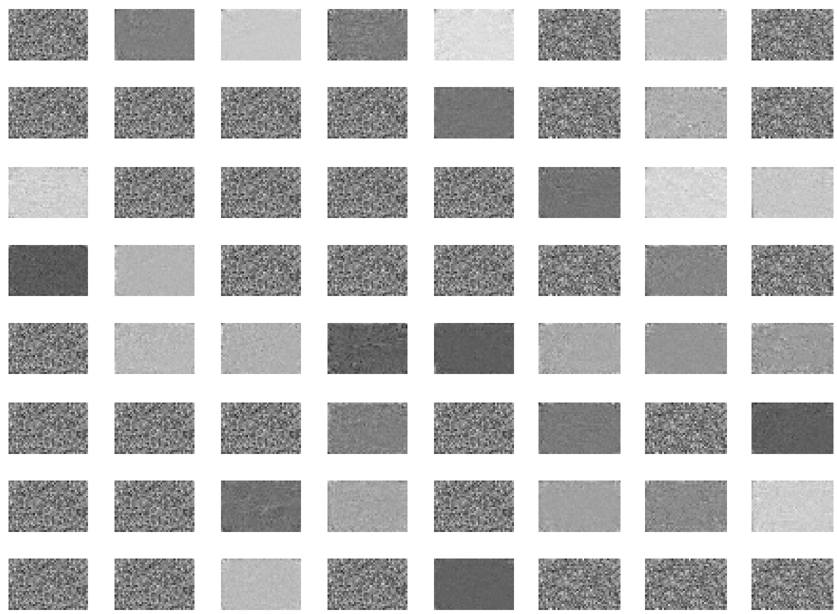
iii)

```
maxact=deepDreamImage(net,i,1:Arch{2,3},'PyramidLevels',1);
```

```
|==============================================|
|  Iteration  |  Activation  |  Pyramid Level  |
|             |   Strength   |                 |
|==============================================|
|           1 |         0.34 |               1 |
|           2 |         0.14 |               1 |
|           3 |         0.52 |               1 |
|           4 |         0.58 |               1 |
|           5 |         0.06 |               1 |
|           6 |         0.87 |               1 |
|           7 |         1.33 |               1 |
|           8 |         1.84 |               1 |
|           9 |         2.14 |               1 |
|          10 |         2.99 |               1 |
|==============================================|
```

```
a=ceil(sqrt(size(maxact,4)));
figure
for j=1:size(maxact,4)
    subplot(a,a,j)
    imagesc(maxact(:,:,1,j))
    axis off
    colormap gray
end
suptitle(['Results from Layer Number ' num2str(2)])
```

Results from Layer Number 2



In this case, maximum activations look smoother than those of MNIST. The reason could be that MNIST is trained with images with a number in the center and there are only 10 categories there. But here the network is trained witha wide range of images from 1000 categories.

iv)

```
Size=224;
categories=dir('101_ObjectCategories');
categories(1:2)=[];
Result=cell(numel(categories),2);
for k=1:numel(categories)
    Directory=categories(k).name;
    im=imread([cd '\101_ObjectCategories\' Directory '\image_0001.jpg']);
    %dealing with grayscale images
    if size(im,3)==1
        im=repmat(im,1,1,3);
    end
    im=imresize(im,[Size Size]);
    label = classify(net, im);
    Result{k,1}=Directory;
    Result{k,2}=char(label);
end
Result
```

```
Result = 101×2 cell array

    {'Faces'          }    {'library'          }
    {'Faces_easy'     }    {'coffee mug'       }
    {'Leopards'       }    {'cheetah'          }
    {'Motorbikes'     }    {'moped'            }
    {'accordion'      }    {'accordion'        }
    {'airplanes'      }    {'warplane'         }
```

{'anchor' }	{'nematode' }
{'ant' }	{'ant' }
{'barrel' }	{'barrel' }
{'bass' }	{'goldfish' }
{'beaver' }	{'kite' }
{'binocular' }	{'binoculars' }
{'bonsai' }	{'snowmobile' }
{'brain' }	{'web site' }
{'brontosaurus' }	{'water buffalo' }
{'buddha' }	{'tile roof' }
{'butterfly' }	{'monarch' }
{'camera' }	{'reflex camera' }
{'cannon' }	{'thresher' }
{'car_side' }	{'cab' }
{'ceiling_fan' }	{'broom' }
{'cellphone' }	{'digital clock' }
{'chair' }	{'rocking chair' }
{'chandelier' }	{'drilling platform' }
{'cougar_body' }	{'cougar' }
{'cougar_face' }	{'cougar' }
{'crab' }	{'rock crab' }
{'crayfish' }	{'isopod' }
{'crocodile' }	{'African crocodile' }
{'crocodile_head' }	{'African crocodile' }
{'cup' }	{'cup' }
{'dalmatian' }	{'dalmatian' }
{'dollar_bill' }	{'book jacket' }
{'dolphin' }	{'jay' }
{'dragonfly' }	{'dragonfly' }
{'electric_guitar'}	{'electric guitar' }
{'elephant' }	{'tusker' }
{'emu' }	{'llama' }
{'euphonium' }	{'cornet' }
{'ewer' }	{'pitcher' }
{'ferry' }	{'catamaran' }
{'flamingo' }	{'hook' }
{'flamingo_head' }	{'flamingo' }
{'garfield' }	{'corkscrew' }
{'gerenuk' }	{'impala' }
{'gramophone' }	{'dial telephone' }
{'grand_piano' }	{'television' }
{'hawksbill' }	{'loggerhead' }
{'headphone' }	{'microphone' }
{'hedgehog' }	{'porcupine' }
{'helicopter' }	{'aircraft carrier' }
{'ibis' }	{'black stork' }
{'inline_skate' }	{'lighter' }
{'joshua_tree' }	{'barn' }
{'kangaroo' }	{'gazelle' }
{'ketch' }	{'yawl' }
{'lamp' }	{'table lamp' }
{'laptop' }	{'notebook' }
{'llama' }	{'llama' }
{'lobster' }	{'knot' }
{'lotus' }	{'ladybug' }
{'mandolin' }	{'pick' }
{'mayfly' }	{'grasshopper' }
{'menorah' }	{'chime' }
{'metronome' }	{'guillotine' }
{'minaret' }	{'slide rule' }

```
{'nautilus'       }    {'chambered nautilus'}
{'octopus'        }    {'letter opener'     }
{'okapi'          }    {'sorrel'            }
{'pagoda'         }    {'sundial'           }
{'panda'          }    {'giant panda'       }
{'pigeon'         }    {'black grouse'      }
{'pizza'          }    {'pizza'             }
{'platypus'       }    {'platypus'          }
{'pyramid'        }    {'barn'              }
{'revolver'       }    {'revolver'          }
{'rhino'          }    {'warthog'           }
{'rooster'        }    {'cock'              }
{'saxophone'      }    {'sax'               }
{'schooner'       }    {'schooner'          }
{'scissors'       }    {'hook'              }
{'scorpion'       }    {'scorpion'          }
{'sea_horse'      }    {'banded gecko'      }
{'snoopy'         }    {'handkerchief'      }
{'soccer_ball'    }    {'soccer ball'       }
{'stapler'        }    {'hair slide'        }
{'starfish'       }    {'sea urchin'        }
{'stegosaurus'    }    {'triceratops'       }
{'stop_sign'      }    {'street sign'       }
{'strawberry'     }    {'strawberry'        }
{'sunflower'      }    {'daisy'             }
{'tick'           }    {'tick'              }
{'trilobite'      }    {'trilobite'         }
{'umbrella'       }    {'umbrella'          }
{'watch'          }    {'whistle'           }
{'water_lilly'    }    {'daisy'             }
{'wheelchair'     }    {'tricycle'          }
{'wild_cat'       }    {'leopard'           }
{'windsor_chair'  }    {'rocking chair'     }
{'wrench'         }    {'hammer'            }
{'yin_yang'       }    {'mouse'             }
```

Complete Result is presented at the end. In my opinon, VGG16 is performing well on Caltech101. Many categories are recognized exactly and others are categorized as objects which have similarities to actual objects. In some cases the image is categorized as other objects present in the image, for example face is categorized as library because there are some books on a shelf in that image.

Preparing data to use as input in part v

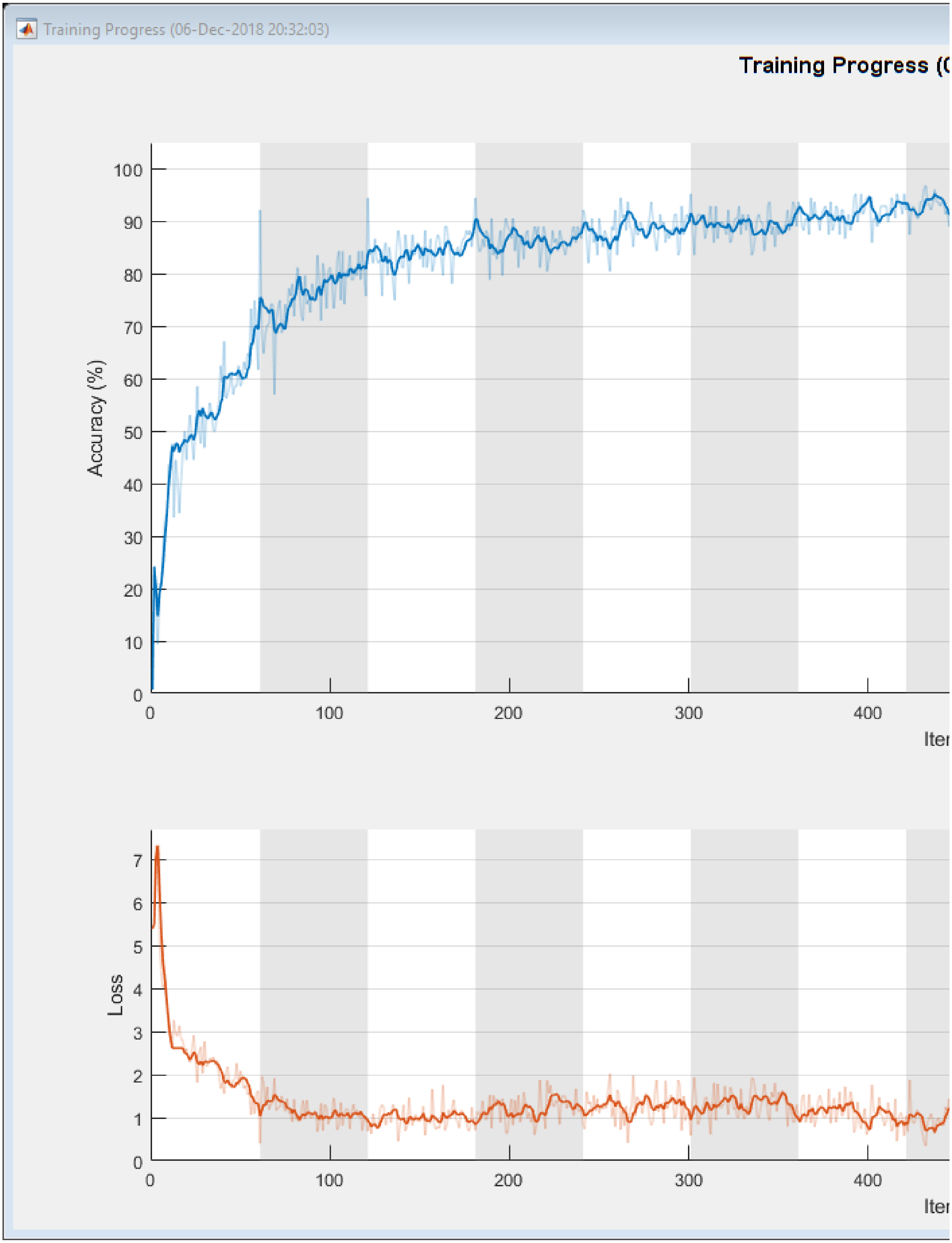
```
Size=224;
categories=dir('101_ObjectCategories');
categories(1:2)=[];
imgDataTrain=[];
labelsTrain=[];
imgDataTest=[];
labelsTest=[];
for k=1:numel(categories)
    Directory=categories(k).name;
    Names1=dir(['101_ObjectCategories\' Directory '\']);
    Names1(1:2)=[];
    for i=1:floor(numel(Names1)*.9)
        im=imread([cd '\101_ObjectCategories\' Directory '\' Names1(i).name]);
        %dealing with grayscale images
        if size(im,3)==1
           im=repmat(im,1,1,3);
```

```
        end
        im=imresize(im,[Size Size]);
        imgDataTrain=cat(4,imgDataTrain,im);
    end
    labelsTrain=[labelsTrain;k*ones(i,1)];
    labelsTest=[labelsTest;k*ones(numel(Names1)-i,1)];
    for j=i+1:numel(Names1)
        im=imread([cd '\101_ObjectCategories\' Directory '\' Names1(j).name]);
        %dealing with grayscale images
        if size(im,3)==1
           im=repmat(im,1,1,3);
        end
        im=imresize(im,[Size Size]);
        imgDataTest=cat(4,imgDataTest,im);
    end
end
labelsTrain=categorical(labelsTrain);
labelsTest=categorical(labelsTest);
```

v)

```
load('D0Data.mat')
net = vgg16;
% defining layers of CNN
for i=1:numel(net.Layers)-1
    layers(i,1)=net.Layers(i);
end
for i=[2 4 7 9 12 14 16 19 21 23 26 28 30]
    layers(i,1).BiasLearnRateFactor=0;
    layers(i,1).BiasL2Factor=0;
    layers(i,1).WeightLearnRateFactor=0;
    layers(i,1).WeightL2Factor=0;
end
layers(39,1)=fullyConnectedLayer(101,'Name','fc8','WeightL2Factor',0);
layers(41,1)=classificationLayer('Name','output');

options = trainingOptions('adam',...
    'MaxEpochs',15,...
    'Plots','training-progress');
clearvars net
net = trainNetwork(imgDataTrain, labelsTrain, layers, options);
```



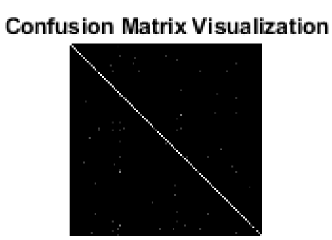

```
Training on single CPU.
Initializing image normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
```

1	1	00:00:25	0.78%	5.3913	0.0010
1	50	00:20:26	58.59%	2.0862	0.0010
2	100	00:40:41	81.25%	1.0199	0.0010
3	150	01:00:56	82.81%	0.9138	0.0010
4	200	01:21:11	88.28%	1.1432	0.0010
5	250	01:41:28	90.63%	1.1046	0.0010
5	300	02:01:42	89.84%	1.1764	0.0010
6	350	02:21:57	87.50%	1.6024	0.0010
7	400	02:42:13	94.53%	0.7107	0.0010
8	450	03:02:27	92.97%	1.0612	0.0010
9	500	03:22:42	91.41%	1.2062	0.0010
10	550	03:42:57	96.88%	0.4988	0.0010
10	600	04:03:11	93.75%	0.8965	0.0010
11	650	04:23:26	96.09%	0.5061	0.0010
12	700	04:43:40	95.31%	0.7111	0.0010
13	750	05:03:56	97.66%	0.3364	0.0010
14	800	05:24:18	90.63%	1.2411	0.0010
15	850	05:44:38	96.09%	0.6227	0.0010
15	900	06:05:00	98.44%	0.2491	0.0010

```
predLabelsTest = net.classify(imgDataTest);
testAccuracy = sum(predLabelsTest == labelsTest) / numel(labelsTest)
```

```
testAccuracy = 0.9166
```

```
[x,y]=meshgrid(unique(labelsTest),unique(labelsTest));
Pred=repmat(reshape(predLabelsTest,1,1,[]),numel(unique(labelsTest)),numel(unique(labelsTest)));
Actual=repmat(reshape(labelsTest,1,1,[]),numel(unique(labelsTest)),numel(unique(labelsTest)));
Confusion_Matrix=sum((((Actual==y)+(Pred==x))==2),3);
Confusion_Matrix=Confusion_Matrix./repmat(sum(Confusion_Matrix,2),1,size(Confusion_Matrix,2));
Confusion_Matrix=Confusion_Matrix/max(Confusion_Matrix(:));
figure,
imshow(Confusion_Matrix)
title('Confusion Matrix Visualization')
```




Training Progress (06-Dec-2018 20:32:03)
Training Progress (06-Dec-2018 20:32:03)
Accuracy (%)
Iteration
Loss
Iteration
10
10
Results
Validation accuracy: N/A
Training finished: Reached final iteration
Training Time
Start time: 06-Dec-2018 20:32:03
Elapsed time: 365 min 0 sec
Training Cycle
Epoch: 15 of 15
Iteration: 900 of 900
Iterations per epoch: 60
Maximum iterations: 900
Validation
Frequency: N/A
Patience: N/A
Other Information
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.001
Learn more
Accuracy
Training (smoothed)
Training
Validation
Loss
Training (smoothed)
Training
Validation