Part C:

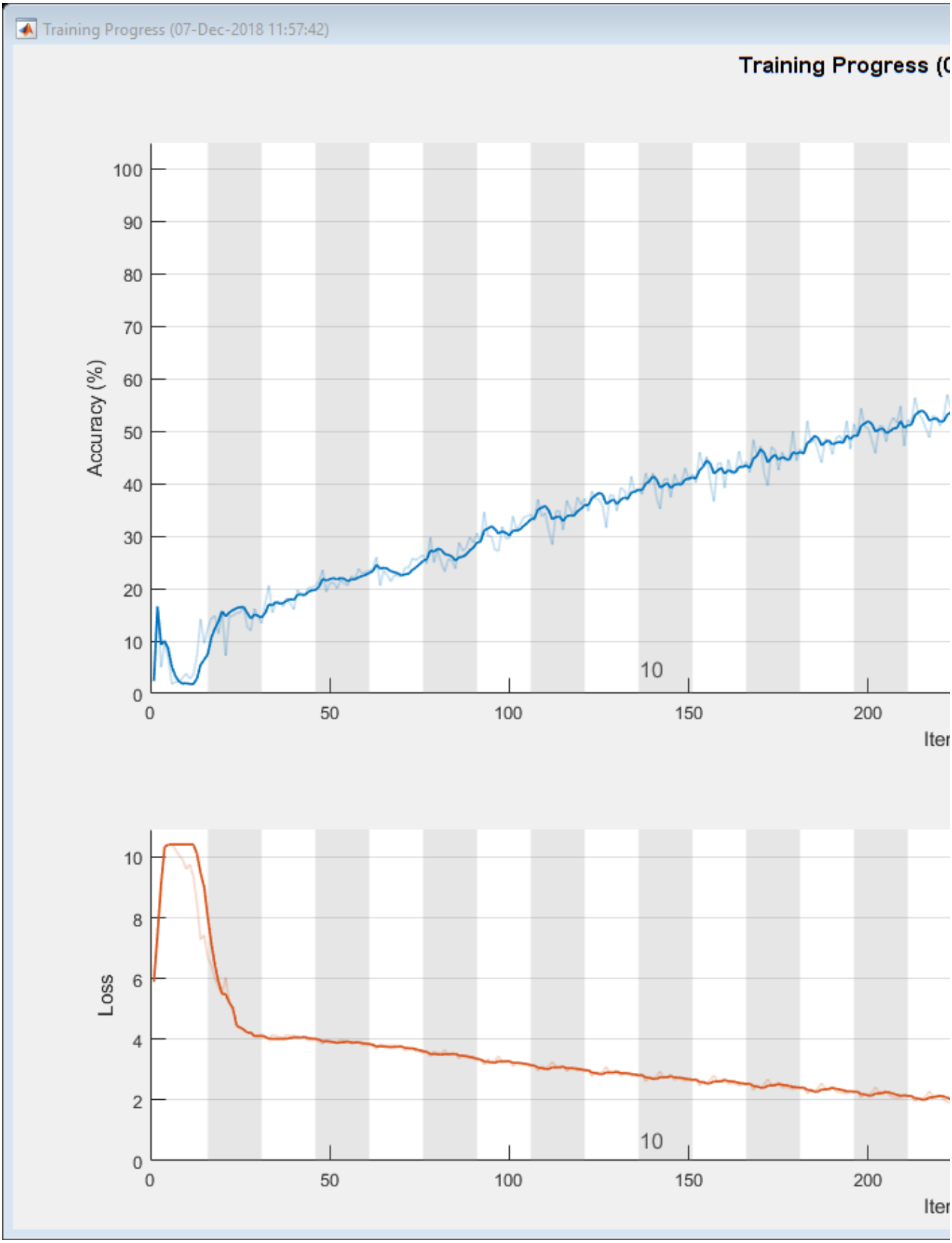
i)Preparing data

```
%size of input images
Size=128;
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,[128 128]);
        imgDataTest=cat(4,imgDataTest,im);
    end
end
labelsTrain=categorical(labelsTrain);
labelsTest=categorical(labelsTest);
```

ii) MNIST implimentation

```
miniBatchSize = 500;
layers = [
    imageInputLayer([128 128 3])
    convolution2dLayer(3,16,'Padding',1)
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(3,32,'Padding',1)
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(3,64,'Padding',1)
    batchNormalizationLayer
```

```
    reluLayer
    fullyConnectedLayer(101)
    softmaxLayer
    classificationLayer];
options = trainingOptions( 'sgdm',...
    'MiniBatchSize', miniBatchSize,...
    'Plots', 'training-progress');
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       |
```

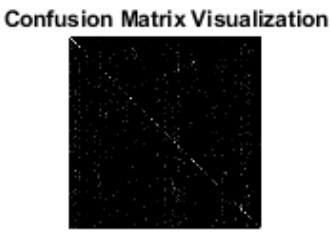
```
|=========================================================================================|
|       1 |           1 |       00:00:07 |        2.40% |       5.8933 |          0.0100 |
|       4 |          50 |       00:05:54 |       21.00% |       3.9361 |          0.0100 |
|       7 |         100 |       00:11:47 |       29.60% |       3.2846 |          0.0100 |
|      10 |         150 |       00:17:41 |       40.80% |       2.6661 |          0.0100 |
|      14 |         200 |       00:23:33 |       50.60% |       2.1592 |          0.0100 |
|      17 |         250 |       00:29:26 |       55.80% |       1.8612 |          0.0100 |
|      20 |         300 |       00:35:18 |       65.20% |       1.4935 |          0.0100 |
|      24 |         350 |       00:41:03 |       75.80% |       1.0033 |          0.0100 |
|      27 |         400 |       00:46:33 |       82.20% |       0.8077 |          0.0100 |
|      30 |         450 |       00:52:02 |       81.00% |       0.7254 |          0.0100 |
|=========================================================================================|
```

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

```
testAccuracy = 0.4226
```

Confusion Matrix Calculation:

```
[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')
```

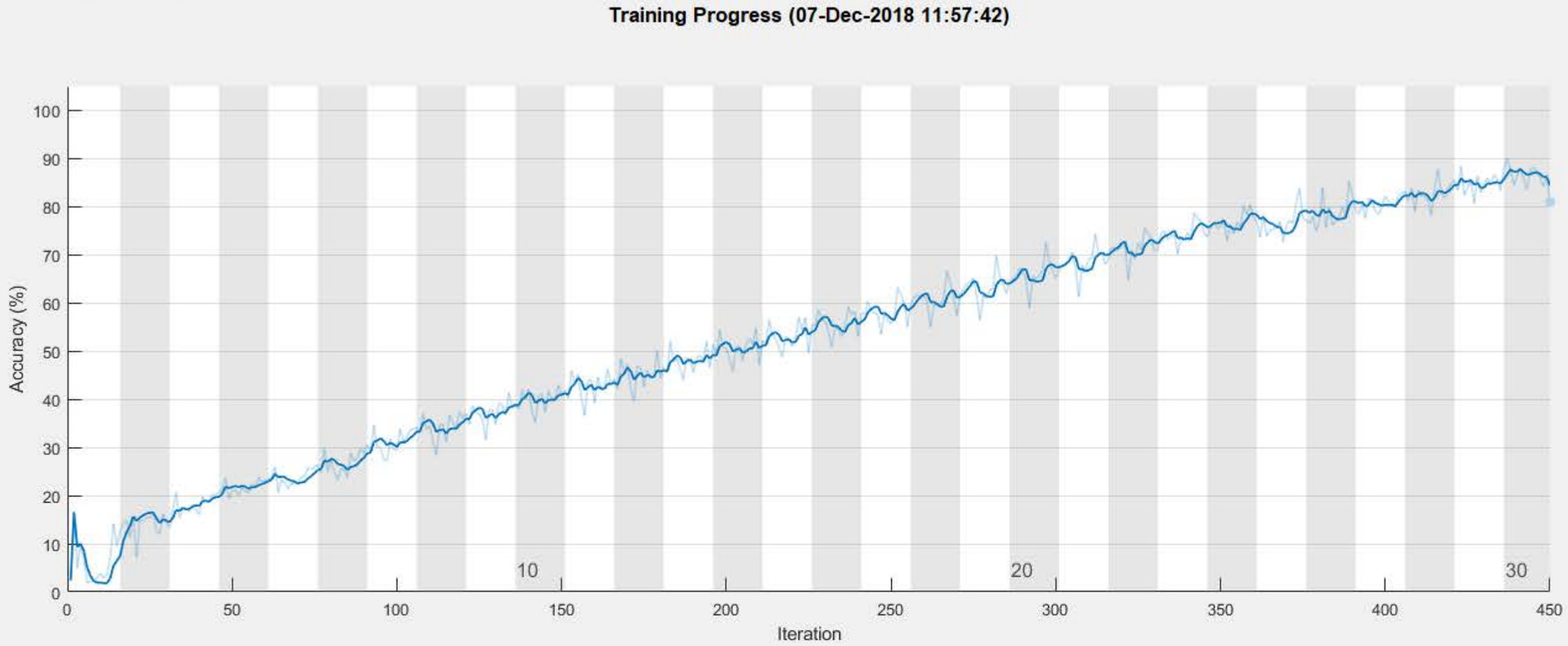


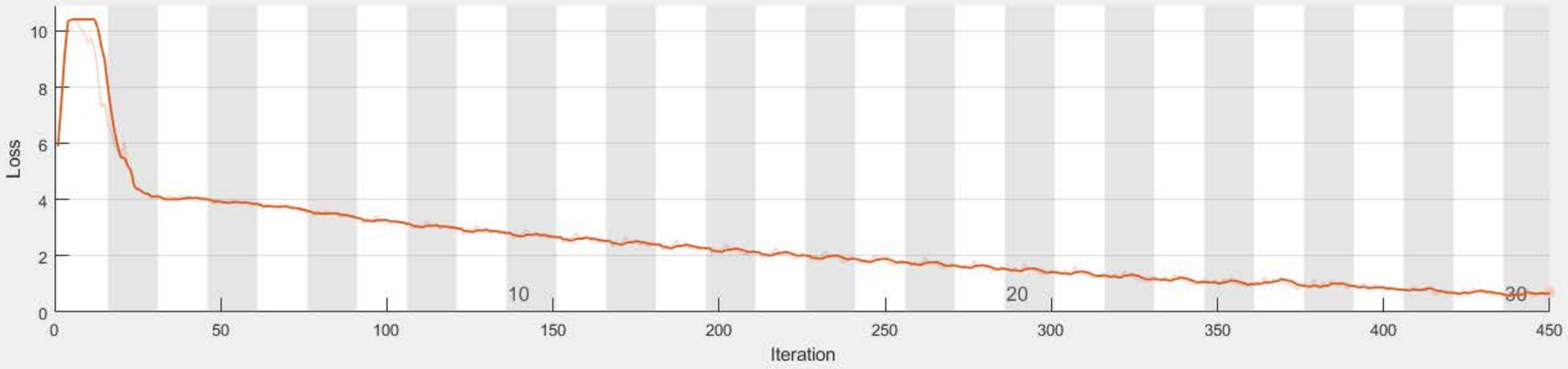
Accuracies reported for Caltech101 range from 15 to 65 percent. Accurace thet I got here (42 percent) is near average. I think the reason is that MNIST is designed for small grayscale images including numbers (and characters) in the center of the image.

It seems that model has been overfitted, because test acuuracy of last epoch is 81 percent.

I tried to train the network with batch size 8000 and it took much longer to run and resulted in 29 percent accuracy.

Training Progress (07-Dec-2018 11:57:42)





Results

Validation accuracy:	N/A
Training finished:	Reached final iteration

Training Time

Start time:	07-Dec-2018 11:57:42
Elapsed time:	52 min 2 sec

Training Cycle

Epoch:	30 of 30
Iteration:	450 of 450
Iterations per epoch:	15
Maximum iterations:	450

Validation

Frequency:	N/A
Patience:	N/A

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.01

Learn more

Accuracy

Training (smoothed)
Training
Validation

Loss

Training (smoothed)
Training
Validation