Home Assignment 3 report

According to my **student ID 1095709**, I have selected **CIFAR100** image dataset and **AlexNet** as my deep convolutional neural network (DCNN) for training it. This experiment has run over **a single GPU(Nvidia GEForce MX150 4GB), 8GB RAM and 4 core CPU** in **MATLAB** environment on my regular laptop.

Implementation:

The CIFAR100 image dataset consists of 60000 32x32 color images in 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. As, this data is in a 3072 features format, with 1024 features for each R, G, B values. I generated CIFAR100 images out of them to take less time for running a program each time and avoided the data array conversion timing which is significantly higher than the image conversion timing.

*In **Alex** **Krizhevsky**, **Ilya** **Sutskever**, **Geoffery Hinton**'s paper, they described the AlexNet architecture as below:*

The first convolutional layer filters the 224×224×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in a kernel map). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size 5 × 5 × 48. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size 3 × 3 × 256 connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size 3 × 3 × 192, and the fifth convolutional layer has 256 kernels of size 3 × 3 × 192. The fully connected layers have 4096 neurons each [1].

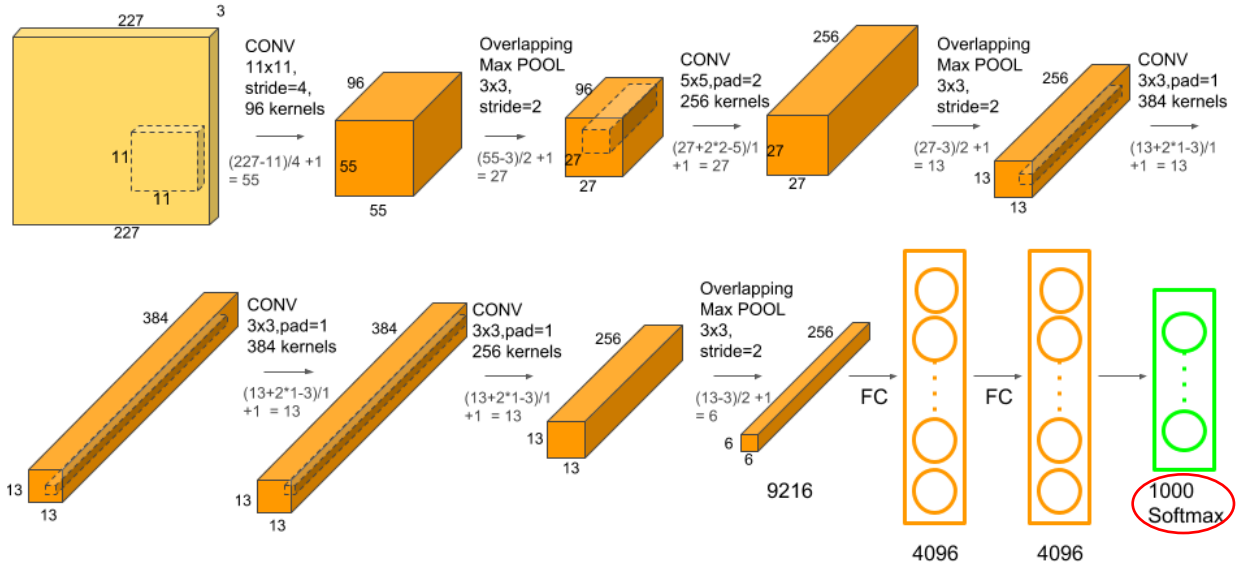

Figure 1: AlexNet architecture

I ran the pre-trained AlexNet DCNN from Figure 1 with 25 layers (5 convolutional followed by RELU and max-pooling layers and 3 fully connected layers followed by combination of dropout, RELU and SoftMax

layer) over 5 epochs. To do so, I **installed AlexNet package** on MATLAB software from Deep Learning Toolbox.

In AlexNet transfer learning, the number of classes to be predicted are 100 in our dataset, so I modified the last fully connected (FC) layer with *OutputSize* parameter as 100, which is predefined as 1000 in AlexNet model as AlexNet has been pretrained on ImageNet dataset with 1000 classes of images, and replaced the existing classification layer with new classification layer having *auto* output parameter.

The CIFAR100 image dataset is of 32x32 color images and AlexNet takes the data input as 227x227 colors images. We used *augmentedImageDatastore()* to convert the dataset 32x32 images into the required 227x227 format in a 4D array data. By providing this augmented image datastore(227x227x3x50000) as input to pre-trained network through trainNetwork() MATLAB function, I got around **71.60% average accuracy (71.46+71.54+71.81/3=71.60%)** for running the pre-trained network 3 times **over 5 epochs** with the *MiniBatchSize* of 64 and *InitialLearnRate* of $3e^{-4}$.

Below are the screenshots for 3 runs for pre-trained AlexNet model.

```
Command Window
  Accuracy for pretrained network
      0.7181

  Time required for training a network
     8.2586e+03
```

Figure 2: Pre-trained network command line output with maximum accuracy

Here you can see that this pre-trained network took around **8.2586e+03 seconds** for training a pretrained network with best accuracy.

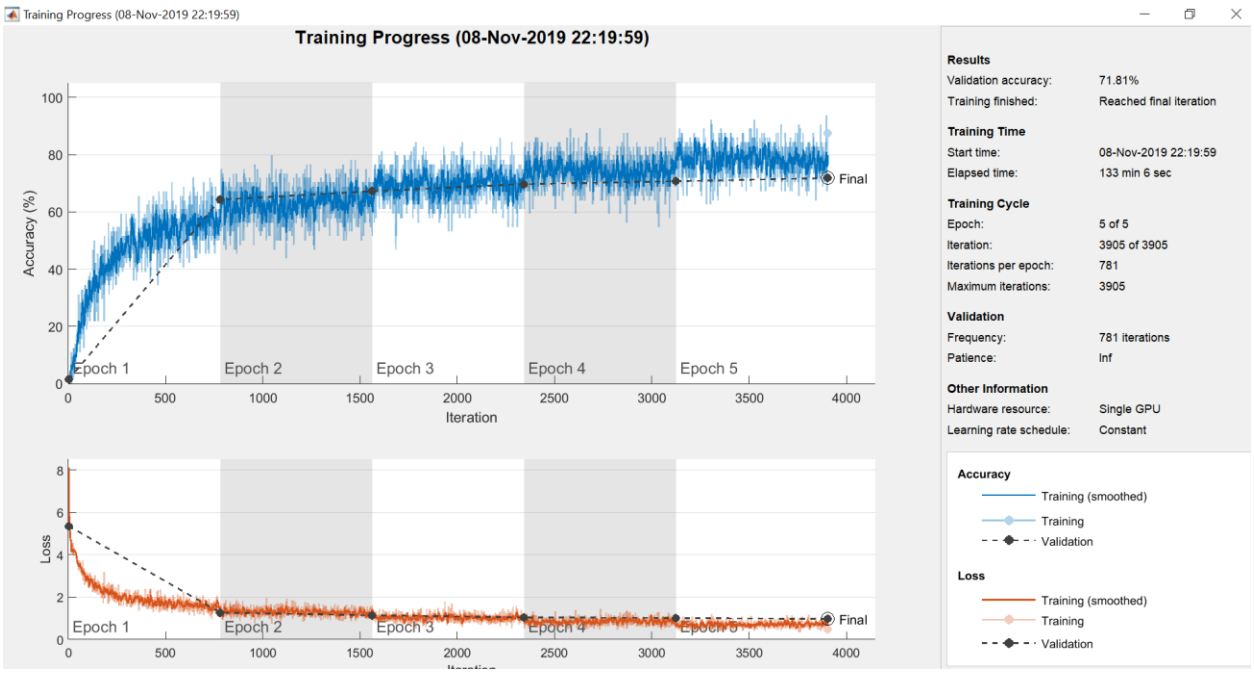

Figure 3: Training progress for pre-trained AlexNet model in run 1.

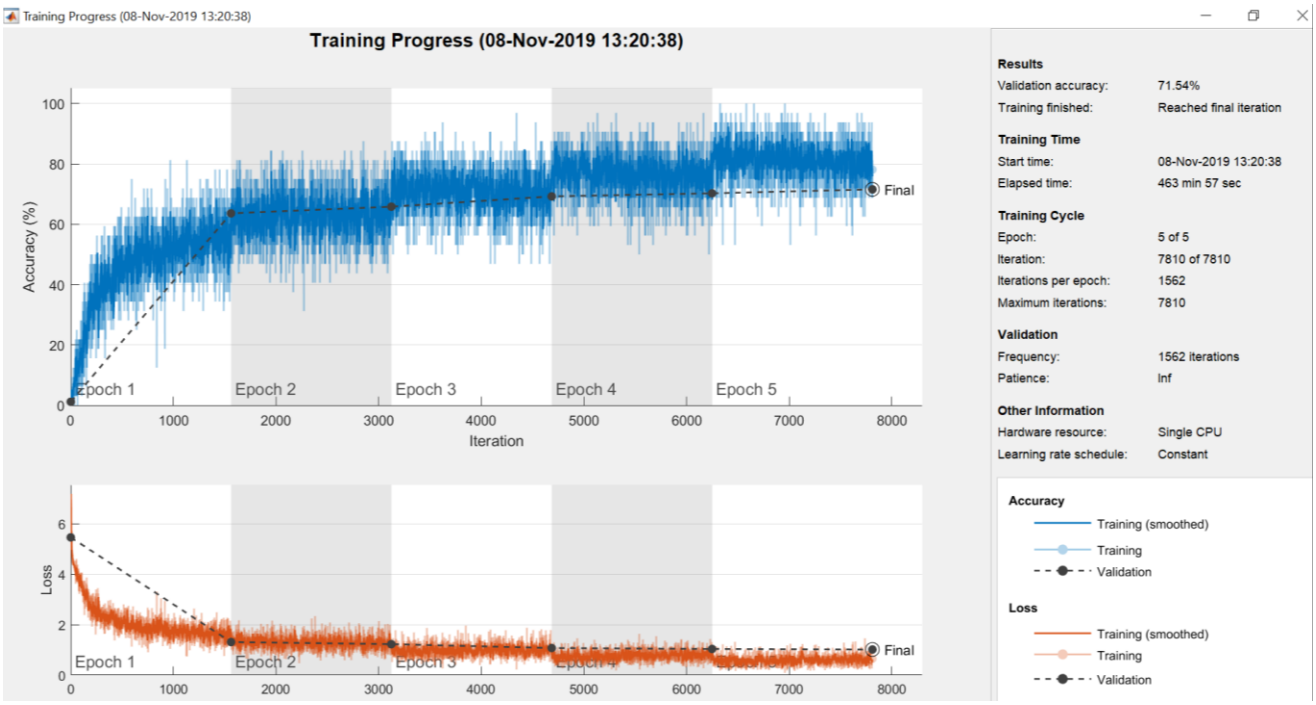

Figure 4: Training progress for pre-trained AlexNet model in run 2.

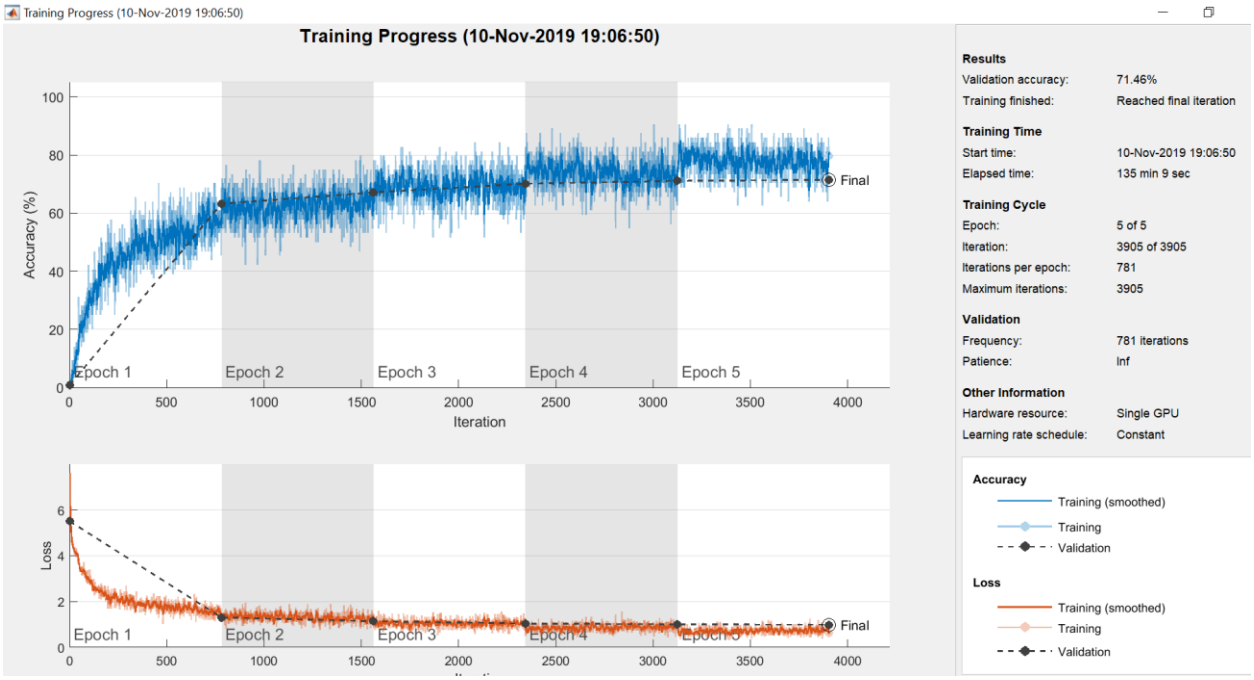

Figure 5: Training progress for pre-trained AlexNet model in run 3.

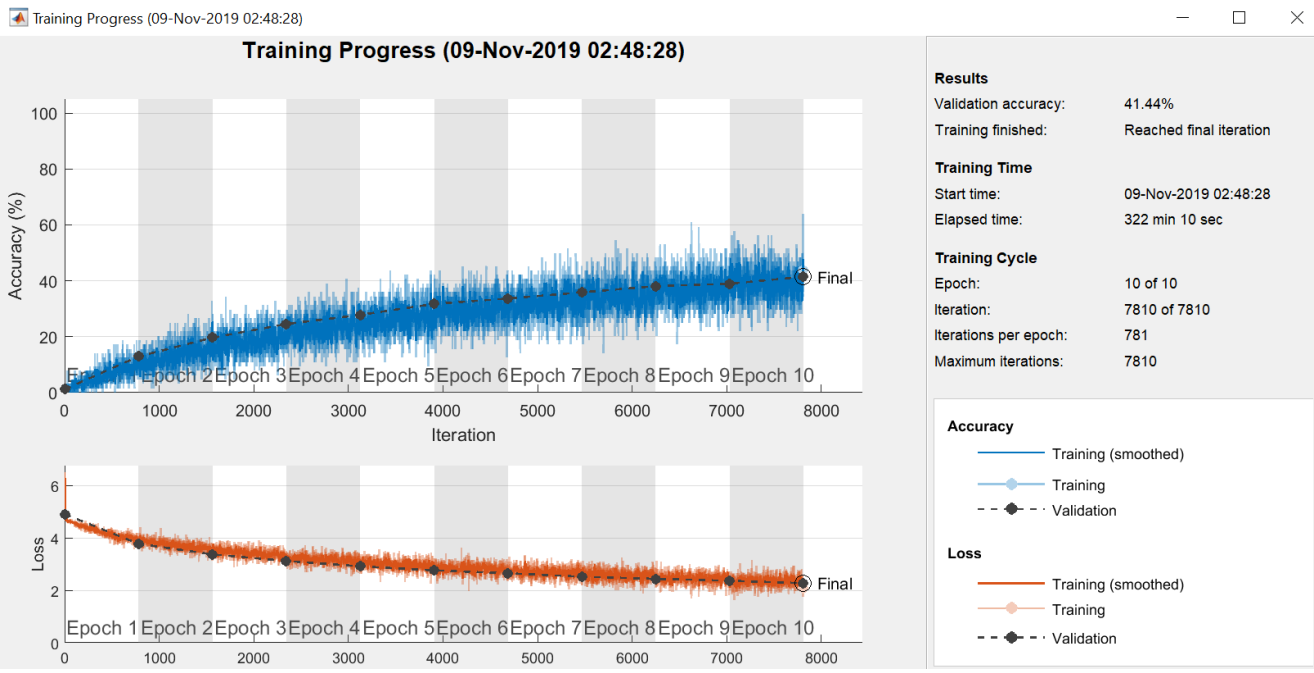

Figure 6: Training progress for learning from scratch AlexNet model in run 1.

```
Command Window
>> deepNetworkDesigner
>> alexScratch
Accuracy for a network from scratch
    0.4144

Time required for training an alexnet network
   2.0449e+04
```

Figure 7: Learning from scratch model command line output

In learning from scratch model, I implemented layers array similar to AlexNet layers and I got around **41.44% accuracy** with **10 epochs** and it took around **2.0449e+04 seconds**.

Analysis and discussion:

From these results, you can see that since, the pre-trained models have calibrated weights already, we are getting **pre-trained model's performance significantly higher than a network being trained from scratch**.

As, I ran the pre-trained network for over 5 epochs it took me around **133 mins.** Whereas, for scratch learning model, it took me around **322mins** to reach at least more than **40 % accuracy in 10 epochs**. At 5th epoch, scratch network model was at around 35% validation accuracy(which is just half of the pre-trained network's accuracy), it's not harmful to consider that to reach at 35% accuracy at 5th epoch, the scratch model might have taken the half the time of full 10 epochs that is **around 322/2=161mins of training time** which is still **more than the training time for pre-trained model**.

From this experiment, we can say that, if we increase the epochs over time, we can **achieve more accuracy over some period**. In learning from scratch, as the weights are not calibrated beforehand, it will take more time and accuracy will increase little by little. However, in pre-trained learning, as the weights are calibrated already for humongous ImageNet dataset over millions of images, the network accuracy grows exponentially over small amount of time and less epochs.

If I obtain high computational power, I think I can improve an accuracy with reduced training time and more number of epochs.

Steps to run pre-trained and scratch AlexNet model:

1. Open pre-trained AlexNet model file named 'alexnetworkMatlab.m' in MATLAB. And just run it without any parameters.
2. Open learning from scratch model file named 'alexScratch.m' in MATLAB. And just run it without any parameters.
3. These both files will open the training progress graph for you to monitor the progress for each epoch.

References:

[1] Alex Krizhevsky, Ilya Sutskever, Geoffery Hinton, "ImageNet Classification with Deep Convolutional Neural Networks".