Assignment Report
Deep Learning (IDML730E)

Pranjal Paliwal - RIT2015015
Ajay Agarwal - RIT2015021

Assignment topic
Study and analyse a Convolution Neural Network for the recognition of handwritten characters.

Our Problem
Recognition of Devanagari Characters (Alphabets and digits) using Convolutional Neural Network.

Introduction
In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.
Since this assignment was aimed at getting hands on experience of different concepts of CNN, we have tried to achieve the goal of this assignment through designing neural networks for character recognition of the Devanagari Characters. We then did a brief comparison of performance of both the networks and benchmarked it with currently achieved results.

Architecture
For the purpose of classification of characters written in Devanagari, we tried to make multiple networks and benchmark their results. We have performed the experimentation using a couple of architectures besides we have also experimented using different strides. We obtained a dataset for Devanagri Characters from Kaggle [1].

About the Dataset:
The dataset consists of 92000 rows (training + validation + testing sample images), and 1025 columns. Each row contains the pixel data ("pixel0001" to "pixel1024"), in greyscale values (0 to 255). The first column represents the Devanagari character class corresponding to each image.

First Network

Since the problem of recognising the Devanagari characters was similar to the character recognition in the MNIST Dataset, we tried to use a network similar to what was proposed by Yan LeCun in LeNet [2].

We made a network with 2 convolutional layers, dropout layer and a 2 fully connected layers. For activation, we used ReLu as the activation function.

The PyTorch Model Summary for our Network can be found below.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 46, 28, 28]           1,196
            Conv2d-2          [-1, 40, 10, 10]          46,040
         Dropout2d-3          [-1, 40, 10, 10]               0
            Linear-4                [-1, 100]         100,100
            Linear-5                 [-1, 46]           4,646
================================================================
Total params: 151,982
Trainable params: 151,982
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.34
Params size (MB): 0.58
Estimated Total Size (MB): 0.92
----------------------------------------------------------------
```

Second Network

Since our first network was able to reach a accuracy of only 97% on Validation Dataset, much below the current benchmark of MNIST at 99.7 % [3], we came to a conclusion that some useful features are not being learnt by our network. Thus, we made a more Deeper Neural Network with more convolutional layers to capture those features.

This network has, 4 convolutional layers, and 4 fully connected layers and uses ReLU as activation function.

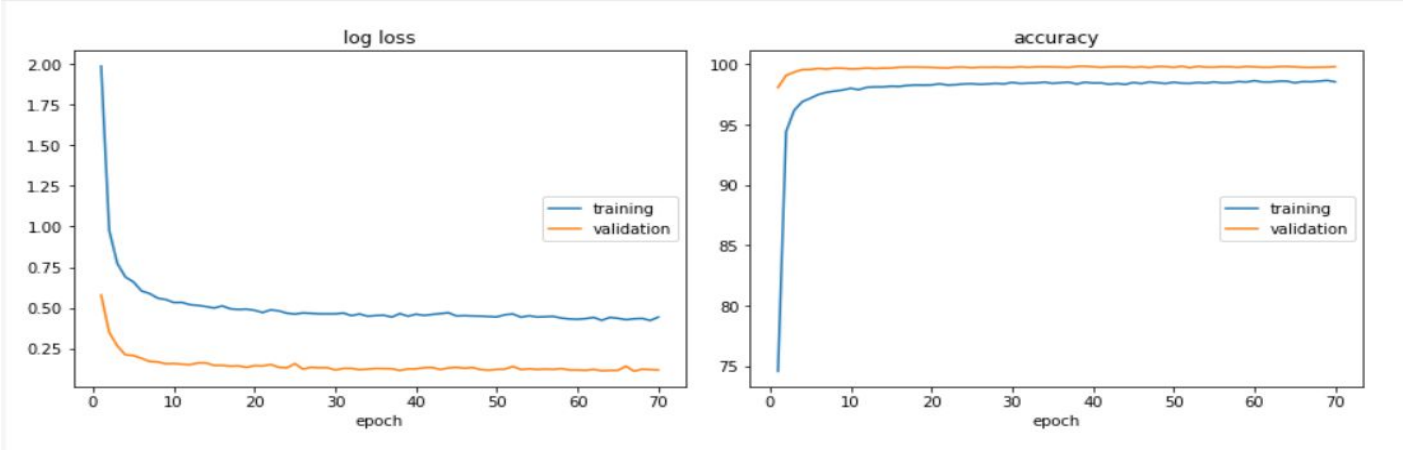
The PyTorch model summary is shown below.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 30, 30]             320
            Conv2d-2          [-1, 64, 28, 28]          18,496
            Conv2d-3         [-1, 128, 12, 12]          73,856
            Conv2d-4         [-1, 128, 10, 10]         147,584
            Linear-5                [-1, 1600]       5,121,600
            Linear-6                 [-1, 128]         204,928
            Linear-7                  [-1, 64]           8,256
            Linear-8                  [-1, 46]           2,990
================================================================
Total params: 5,578,030
Trainable params: 5,578,030
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.85
Params size (MB): 21.28
Estimated Total Size (MB): 22.14
----------------------------------------------------------------
```

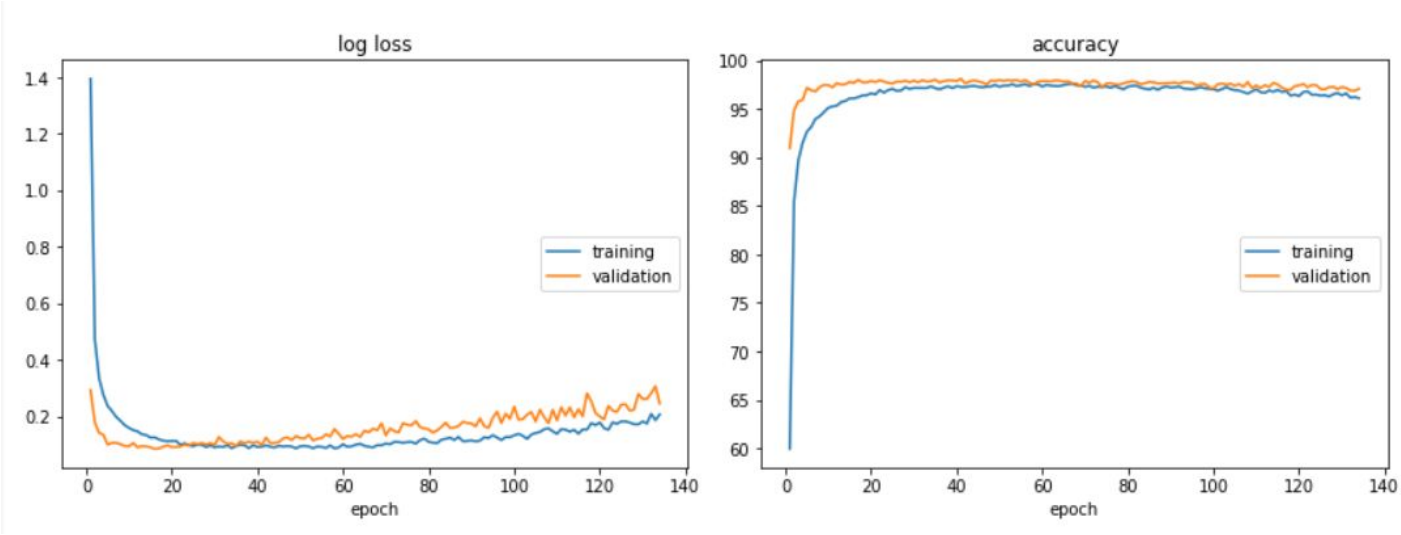
Experiments and Analysis

We conducted various experiments on both the models and tried to find out top-1 as well as top-5 detection accuracy on validation dataset. Following are the accuracies and graphs for various experiments.

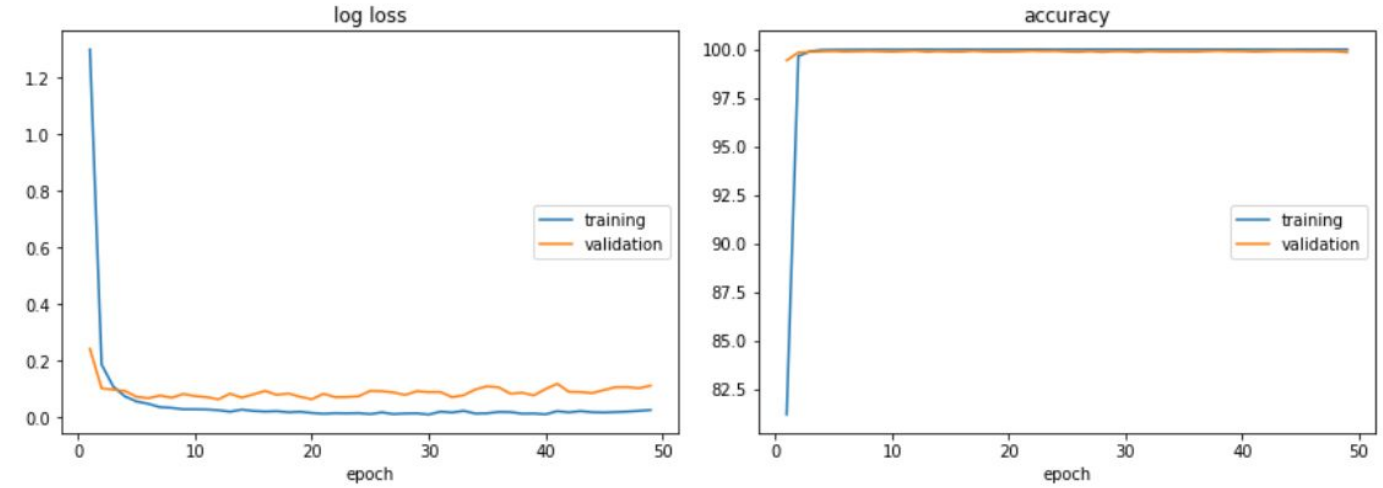
Network 1 Loss and Accuracy: Top-1 Accuracy: **97.05%** Top-5 Accuracy: **99.73%**
We can see that there is a gap between training and validation loss values. This is due to using a couple of Conv2D Dropout layers which are activated during training but act as convolutional layer during validation pass. For further reading please refer this.



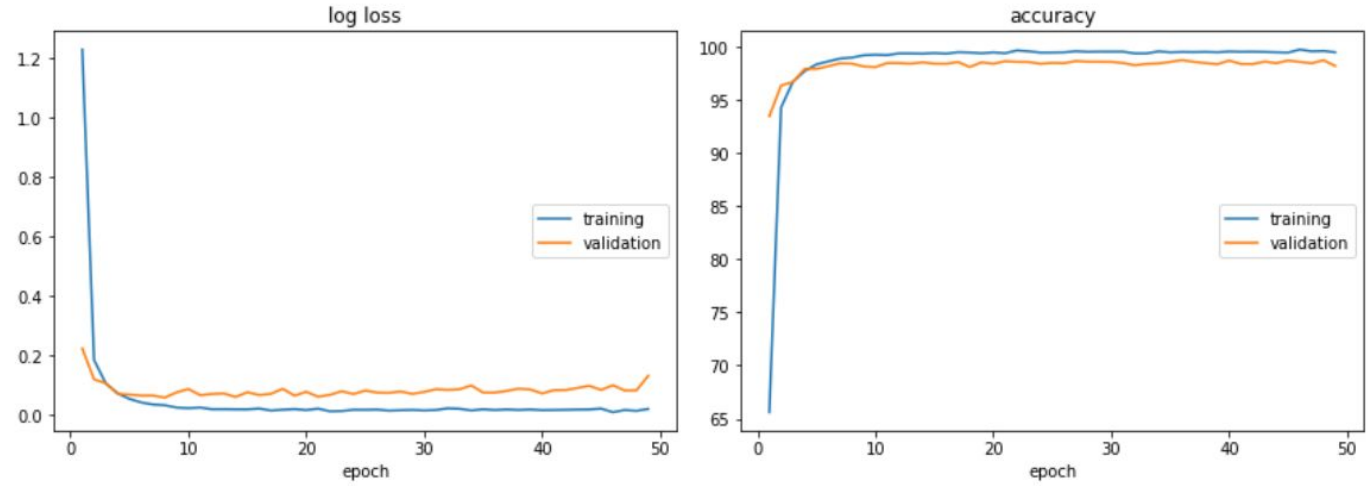
Network 1 - Overtraining Case: Due to overtraining, we can observe that for loss has started to increase in subsequent epochs and accuracy has started to fall off. This could be justified since we have used a higher value of momentum (0.9) and learning rate (0.05) while training. This case helps us understand the need for early-stopping [4].



Network - 2 Top-5 Accuracy and Loss Plot: Top-5 Accuracy after 50 epochs: **99.85%**
As we stated earlier that our network 2 is deep enough to capture most of the features, hence, we can see that the top - 5 accuracy is much greater than that of previous model's.



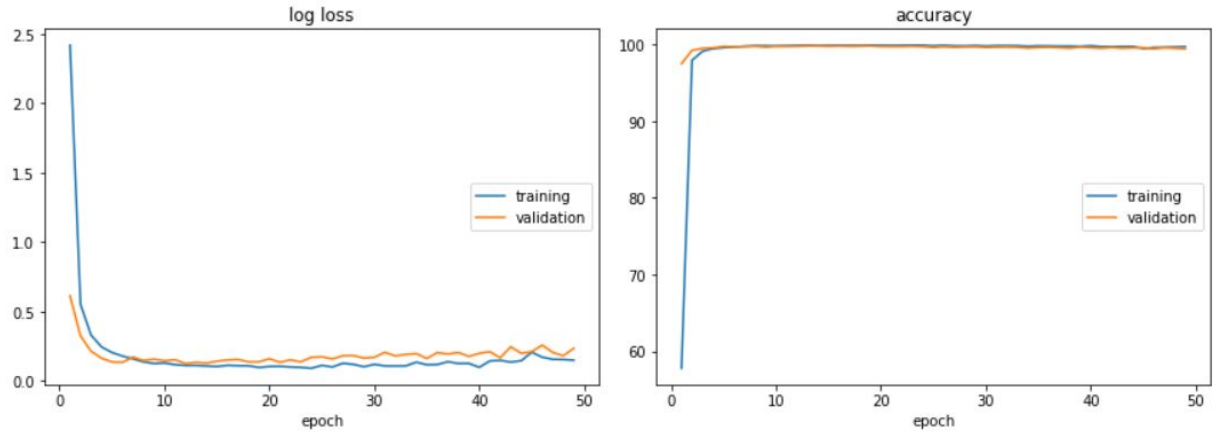
Top-1 Accuracy and Loss Plot for Network - 2 : Final Top-1 Accuracy after 50 epochs: **98.17%**
Since model - 2 captures more features as compared to model - 1, hence top-1 accuracy is more than that of model-1.



Experiments with Stride

1. Stride = 3 in First Max Pooling Layer. Earlier stride was equal to 2. We observed a significant reduction in parameter size on decreasing stride with a corresponding tradeoff in Top-1 Accuracy which is reduced from 98.17% after epochs in Stride = 2 to 96.41 in Stride = 3.

   Below is the Plot for Accuracy (Top-5) and Loss for the changed model.



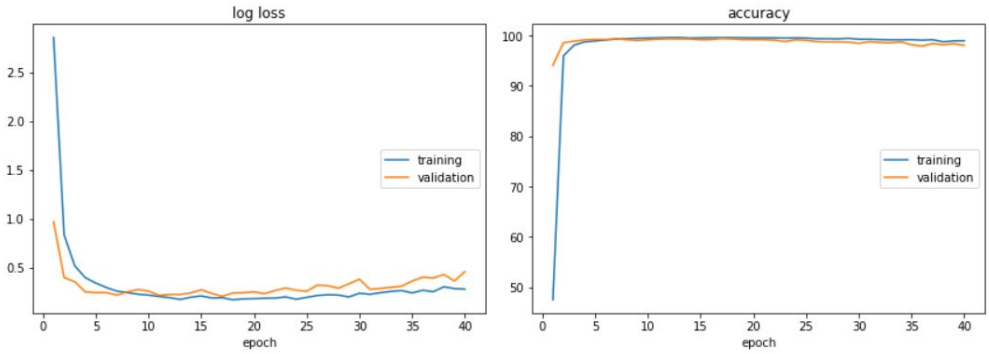
The updated PyTorch model summary is shown below:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 30, 30]             320
            Conv2d-2          [-1, 64, 28, 28]          18,496
            Conv2d-3           [-1, 128, 7, 7]          73,856
            Conv2d-4           [-1, 128, 5, 5]         147,584
            Linear-5                 [-1, 256]         131,328
            Linear-6                 [-1, 128]          32,896
            Linear-7                  [-1, 64]           8,256
            Linear-8                  [-1, 46]           2,990
================================================================
Total params: 415,726
Trainable params: 415,726
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.68
Params size (MB): 1.59
Estimated Total Size (MB): 2.27
----------------------------------------------------------------
```

You can see the significant reduction in number of parameters from 5,578,030 to 415,726. Correspondingly, the network learnt significantly faster than with stride as 2.

2. Stride = 4 in first Max Pooling layers and Stride = 3 in second Max Pooling Layer. An even greater reduction in number of parameters was observed. However, we can observe that this case does not converge with older learning rate of 0.03 and momentum 0.9. The network fails to capture all the features and is thus seen to achieve a maximum top-1 accuracy of only 93.52 and top-5 accuracy of 98% after training several epochs.

Below is the log loss plot and network summary.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 30, 30]             320
            Conv2d-2          [-1, 64, 28, 28]          18,496
            Conv2d-3           [-1, 128, 5, 5]          73,856
            Conv2d-4           [-1, 128, 3, 3]         147,584
            Linear-5                  [-1, 96]          12,384
            Linear-6                  [-1, 80]           7,760
            Linear-7                  [-1, 64]           5,184
            Linear-8                  [-1, 46]           2,990
================================================================
Total params: 268,574
Trainable params: 268,574
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.64
Params size (MB): 1.02
Estimated Total Size (MB): 1.67
----------------------------------------------------------------
```

We can observe that high reduction in network features is corresponded by significant reduction in accuracy.

Analysis Summary

In this assignment on modelling a CNN, we worked on a couple of networks. Both the networks were used for classification of images of handwritten images of devanagari characters. After performing various experiments with our model, following is the list analysis we have come across:

- The model that we created have achieved the best top-1 accuracy of **98.17%** and best top-5 accuracy of **99.85%**. These accuracies are at par with the latest models for classifying the MNIST dataset which is the similar task as our. Please refer to this link [3] for getting latest accuracy for MNIST dataset.
- As we saw that accuracy of model-1 was less than that of model-2 because of less number of layers in model-1 as compared to model-2, hence, we analyzed that model-1 was missing out on capturing some parameters whereas model-2 captured most of the parameters needed for proper classification. **Hence, using a deeper network helps us capture more parameters.**
- Using a **dropout layer helps in achieving generalization** and hence improves validation accuracy. Although there is an improvement in validation accuracy, we find that training accuracy lags behind the validation accuracy. This is due to the fact that in dropout layer some of the neurons are switched off during training and while validation step dropout layer acts like normal Conv. layer. For further reading please refer to this.

- While training a CNN we should try to follow the notion of **Early Stopping** otherwise we may overtrain the network which will lead to reduction of accuracy and increase in loss values in subsequent epochs.
- We have also experimented using different stride values in our network and we have observed that **on increasing the stride value, we start to loose capturing some features** in the CNN which leads to reduced accuracy which is evident from our experiment above. Also increasing the stride value leads to reduction of training parameters of the model.

Visualization of Results

Below are some results from the model obtained using network 2.

```
Prediction Class:  40
```

Actual Character image from Test Dataset

Reference character image from Mapping obtained using Training Data

```
Prediction Class:  3
```

Actual Character image from Test Dataset

Reference character image from Mapping obtained using Training Data

```
Prediction Class:  12
```

Actual Character image from Test Dataset

Reference character image from Mapping obtained using Training Data

```
Prediction Class:  24
```

Actual Character image from Test Dataset

Reference character image from Mapping obtained using Training Data

References

1. Kaggle Dataset for Devanagari Characters. https://www.kaggle.com/rishianand/devanagari-character-set/version/3
2. LeNet by Yan LeCun, [LeCun et al., 1998] http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
3. Neural Network Classification Benchmarks http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#4d4e495354
4. Early-Stopping for Neural Network training https://en.wikipedia.org/wiki/Early_stopping