ML – Assignment (IDfy, OCR solution)

To solve the problem of recognition of text from license plate images, I have used a neural network with CNN and RNN layers. This implementation is done on pytorch framework using the nn.torch module of pytorch that gives the freedom of building neural networks by adding layers from the package. Below, shows some of the data pre-processing steps along with the model architecture followed by the working and results.

**Data Pre-Processing:**

Once all the images were extracted from the given link for dataset, two folders were created “train” and “test” respectively. All the images were split into 80:20 ratios using the train\_test\_split function on given csv of the dataset. Once the items were known, the images were split into train and test folders. Below shows the required path directory for images,

TrainData = /dataset/Train"

TestData = /dataset/Test"

Data is loaded into pytorch using DataLoders which read in the images as Tensors and further operation on these images will be easy.

**Model Architecture:**

The Model consists of 5 convolution layers where the first 3 layers have kernel size of 5 and the rest 2 have kernel size of 2. These convolution layers take input as the Tensors of images with shape [batch\_size, channels, height, width]. Finally the last convolution layer outputs a feature map of the input image. A Relu activation function layer is added to this which activates the neurons and this is a non-linear function. A maxpooling layer is added to it and finally 2 layers of Bidirectional LSTM layers are added that helps in predicting the sequence in the images.

**Working:**

There is a training loop in pytorch where all the action happens. For each epoch, it iterates over each sample in the dataset here, each image. For each time step a CTC loss is calculated which is a many-to-one classification and it does not consider alignment which is an advantage for us to use it in the problem. This loss is calculated between the prediction\_probabilities and the target tensors. The loss is backpropagated using loss.backward() which helps in updating the weights for better predictions.

**Results:**

Below image shows the graph plotted for train and test loss for 10 epochs,

