**Handwritten Text Recognition**

**Abstract**

This present project approaches the Handwritten Text Recognition problem by recognizing a handwritten line. Nowadays, since we live in a digital era, there are a lot of situations in which the text we have written also needs to be in a digital format.

Most organizations use documents to acquire information from customers. These documents are generally handwritten. Such documents can be forms, checks, etc.

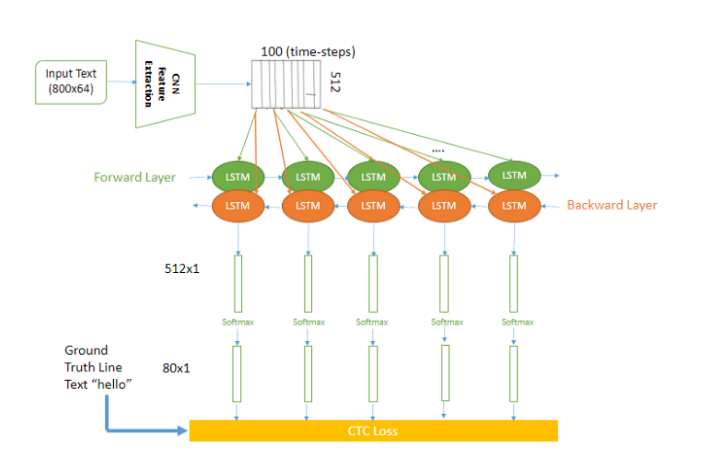
For their easier retrieval or information collection documents are transformed and stored in digital formats. Common practice to handle that information is manually filling same data into computer. It would be tiresome and time consuming to handle such documents manually.

Hence the requirement of a special Handwritten Text Recognition arises.

**Implementation**

For this project we have used the IAM Dataset, which consists of a large handwriting database.

All of the images from the IAM dataset are resized and split into Train and Test sets respecting a given percentage and further split in batches. Along with loading them their ground truth texts (the exact texts that are written in the images) are loaded and further used in the neural network.



First, there are 7 Convolutional Recurrent Neural Network used to extract the relevant features of the images, then the Recurrent Neural Network layers consists of 2 stacked layers of Bidirectional Long Short-Term Memory cells, the output is a matrix which represents the character-scores for each sequence-element.

After that, there comes the Connectionist Temporal Classification, which is in fact an operation that given the RNN output tries all possible alignments of the ground truth text in the image and takes the sum of all scores. While training the NN, the CTC computes the loss value of a batch and during inferring the CTC decodes the matrix from RNN into a text.

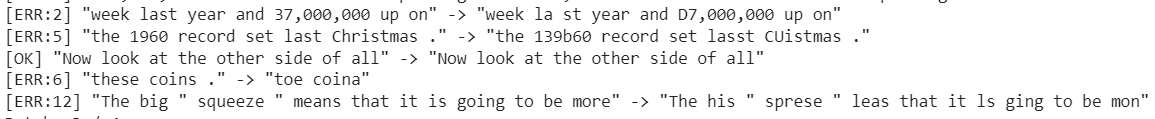
The decoding part uses the method known as Best Path Decoding and just takes the output of RNN and computes an approximation by taking the most likely character at each position.

The train phase takes in each epoch the shuffled train data set and feeds the model one by one with a batch and receives a loss value for it, after each train epoch, a validation is made on the test set where the characters error rate, accuracy and words error rate are calculated, if the character error rate is the best until now the snapshot of the model is saved in a file.

**Results**

To clearly observe the progress of the training, we execute tests after each training epoch. Here we have an example of a recognized entire line in the beginning of the training, among other partially correct lines:

****



For the tests we managed to run we obtained as best result a character error rate of 30.697889% and an accuracy of 1.25%

**Future improvements**

Regarding future improvements, in order to obtain quicker results we only used a small dataset part, so the model should be further trained.

We have obtained a better character recognition rate with each training epoch, but still low values for the accuracy. In each epoch for every line many words are detected correctly and only a small number of characters are misrecognized, resulting in a low accuracy. Other approaches should also be considered in order to make a comparison with our model and an option to input an image and recognize the text from it should be added as a functionality.

**Literature review**

We consulted 2 articles on this subject and based our project on them.

The first article, entitled ***Build a Handwritten Text Recognition System using TensorFlow*** proposesa solution for word recognition which uses a Neural Network (NN), which consists of 5 CNN (Convolutional NN) and 2 RNN (Recurrent NN) layers and outputs a character-probability matrix plus a CTC function for loss calculation and decoding. An implementation using TF (TensorFlow) is provided and some important parts of the code implementation are presented. Both the ground truth text and the recognized text can be at most 32 characters long. As improvement, in order to feed complete text-lines instead of word-images, we need to increase the input size of the neural network.

On the other hand, the article ***How to Make Real-Time Handwritten Text Recognition With Augmentation and Deep Learning*** approaches handwritten line text recognition without pre segmentation into words or characters. It firstly uses Convolutional Recurrent Neural Network to extract the important features from the handwritten line text Image. There are 3 steps for the solution: the Multi-scale feature Extraction, which implies Convolutional Neural Network, Sequence Labeling (BLSTM-CTC) using Recurrent Neural Network, and Transcription, which means Decoding the output of the RNN. To improve accuracy data augmentation is used. Data augmentation reduces line thickness, random noise, blur filter, and random stretch.

**Conclusion**

Handwritten text recognition is a computer vision subject that is worth further improving and analyzing. After we manage to complete the proposed improvements it is worth trying to expand the solution to cover more text, for example to try recognizing a paragraph. Also we could explore optimizing or integrating existent solutions/software that does approximately the same thing.

**References**

[1] Harald Scheidl, [*Build a Handwritten Text Recognition System using TensorFlow*](https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5) (2018)

[2] Sushant Gautam, [*How to Make Real-Time Handwritten Text Recognition With Augmentation and Deep Learning*](https://medium.com/swlh/learn-and-use-handwritten-line-text-recognition-using-deep-learning-with-tensorflow-b661434b5e3b)(2020)