Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

Handwritten Text Recognition using Deep Learning with TensorFlow

Sri. Yugandhar Manchala
Assistant Professor
Computer Science and Engineering
Aditya Institute of Technology and Management
Srikakulam, Andhra Pradesh

Jayaram Kinthali, Kowshik Kotha,
Kanithi Santosh Kumar, Jagilinki Jayalaxmi
Student
Computer Science and Engineering
Aditya Institute of Technology and Management
Srikakulam, Andhra Pradesh

***Abstract*— Character recognition is one in all the emerging fields within the computer vision. The most abilities of humans are they will recognize any object or thing. The hand transcription can easily identify by humans. Different languages have different patterns to spot. Humans can identify the text accurately. The hand transcription cannot be identified by the machine. It's difficult to spot the text by the system. During this text recognition, we process the input image, extraction of features, and classification schema takes place, training of system to acknowledge the text. During this approach, the system is trained to seek out the similarities, and also the differences among various handwritten samples. This application takes the image of a hand transcription and converts it into a digital text.**

Keywords—HTR(Handwritten Text Recognition), NN(Neural Network),CNN(convolutional Neural Network), RNN(Recurrent Neural Network), CTC(Connectionist Temporal Classification), TF(TensorFlow)

I. INTRODUCTION

Image processing could be a manipulation of images within the computer vision. With the event of technology, there are many techniques for the manipulation of the photographs. The text recognition includes a huge role in many areas. But it's difficult to try and do such a task by a machine. For recognition, we've to coach the system to acknowledge the text. The character recognition involves several steps like acquisition, feature extraction, classification, and recognition. Handwriting recognition is the ability of a machine to receive and interpret the handwritten input from an external source like image. the most aim of this project is to style a system that may efficiently recognize the actual character of format employing a neural network. Neural computing could be a comparatively new field, and style components are therefore less well-specified than those of other architecture. Neural computers implement data parallelism. A neural computer is operated in a very way that's completely different from the operation of a standard computer. Neural computers are trained and that they don't seem to be programmed. so that the given data is compared to the trained system and provides the acceptable output text to the user. A handwriting recognition system handles formatting, performs correct segmentation into characters, and finds the foremost plausible words. Off-line handwriting recognition involves the automated conversion of the text in a picture into letter codes that are usable within computer and text-processing applications. the info obtained by this manner is considered a static representation of handwriting.

LITERATURE SURVEY

K. Gaurav, Bhatia, various pre-processing techniques involve within the character recognition with different reasonable images ranges from easy handwritten form-based documents and documents containing colored and complicated background and varying intensities. The offline character recognition is proposed by sing diagonal feature extraction. it's supported the ANN model. There are two approaches for make neural network system such as 54 feature and 69 features. A. Brakensiek, J. Rottland, A. Kosmala, J. Rigoll, during this paper a system for off-line cursive handwriting recognition was described which is based on Hidden Markov Models (HMM) using discrete and hybrid modeling techniques. R. Bajaj, L. Dey, S. Chaudhari, has employed three different styles of features, such as, the density features, moment features, and, descriptive component features for classification of Devanagari Numerals. they obtained 89.6% accuracy for handwritten Devanagari numerals. M. Hanumadhulu and O.V. Ramanammurty have implemented this using a fuzzy set using the box approach and also the recognition is 90%. This model operates on varied information sources. In past researches, it's clear that this model is successful with diverse information sources but it lacks a small amount of accuracy within the case of long sentences. Many proposed models don't seem to be so successful incorrectly classifying the long text data. On the other side models are incorporating CNN networks and showing good results which are because of its capability of dealing the longer text data.

II. PROPOSED WORK

A. *Handwritten text recognition:*

Handwritten Text Recognition (HTR) systems consist of handwritten text in the form of scanned images as shown in figure 1. we are going to build a Neural Network (NN) which is trained on word-images from the IAM dataset. because the input layer (and therefore also all the opposite layers) are often kept small for word-images, NN-training is possible on the CPU (of course, a GPU would be better). For the implementation of HTR, the minimum requirement is TF.

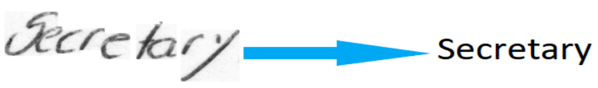


Fig. 1: Image of word taken from IAM Dataset

B. Model Overview:

We use a NN for our task. It consists of a convolutional neural network (CNN) layers, recurrent neural network (RNN) layers, and a final Connectionist Temporal Classification (CTC) layer.

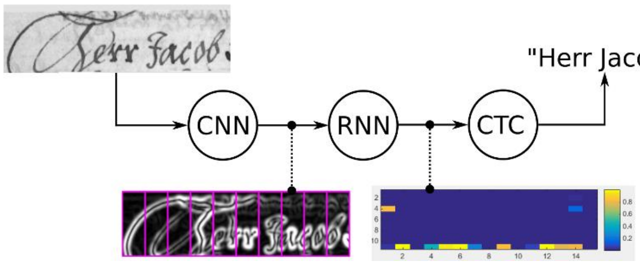


Fig. 2: Overview of HTR

In this project, we've taken 5 CNN (feature extraction) and a pair of RNN layers and a CTC layer (calculate the loss). first, we've to preprocess the pictures in order that we are able to reduce the noise.

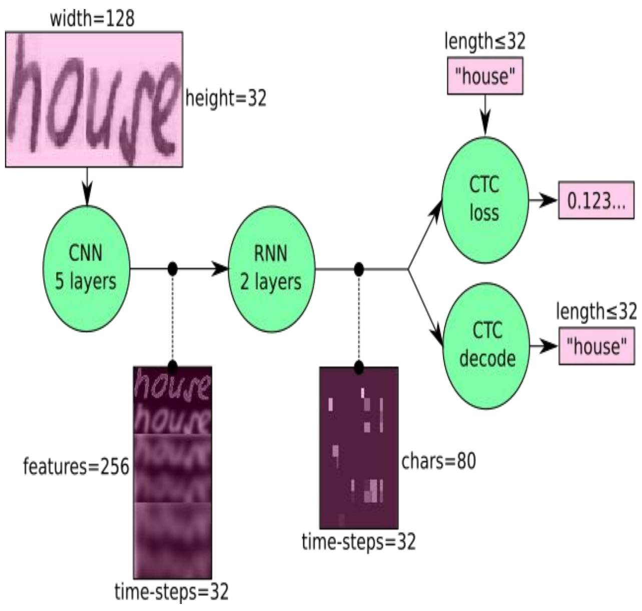


Fig. 3: Green indicates the operations of NN and Pink indicate the dataflow through NN.

We can also view the NN in an exceedingly more formal way as a function (see Eq. 1) which maps a picture (or matrix) M of size W×H to a personality sequence (c1, c2, …) with a length between 0 and L. As you'll see, the text is recognized on character-level, therefore words or texts not contained within the training data is recognized too (as long because the individual characters get correctly classified).

$$\text{NN:}\quad \underset{W \times H}{M} \longrightarrow \underset{0 \leq n \leq L}{(C_1, C_2, \ldots\ldots\ldots, C_n)}$$

Eq. 1: The Neural Network was written as a mathematical function that maps an image M to a character sequence (c1, c2, …).

C. Operations:

CNN: The input image is given to the CNN layers. These layers are trained to take out relevant features from the image. Each layer consists of three operations. First, the convolution operation, 5×5 filter is used in the first two layers and 3×3 filter used in the last three layers to the input. Then, the non-linear RELU function is applied. At last, a pooling layer summarizes image regions and outputs a downsized(smaller) version of the input. While the height of image size is reduced by 2 in each layer, feature channels are added, so that the output feature sequence has a size of 32×256.

RNN: The feature sequence consists of 256 features per time-step, the RNN propagates relevant information through this sequence. The favored Long Short-Term Memory (LSTM) implementation of RNNs is employed because it is in a position to propagate information through longer distances and provides more robust training-characteristics than vanilla RNN. The RNN output sequence is mapped to a matrix of 32×80. The IAM dataset contains 79 different characters, further one additional character is required for the CTC operation (CTC blank label), so that there are 80 entries for every of the 32 time-steps.

CTC: while training the NN, the CTC is given the RNN output matrix and also the ground truth text and it computes the loss value. While inferring, the CTC is just given the matrix and it decodes it into the ultimate text. Both the bottom truth text and also the recognized text are often at the most 32 characters long.

Data:

Input: It is a gray-value image of size 128×32. Usually, the pictures from the dataset don't have exactly this size, therefore we resize it (without distortion) until it either contains a width of 128 or a height of 32. Then, we place the image in a (white) target image of size 128×32. This process is shown in Fig. 3. Finally, we will normalize the gray-values of the image so that it could simplify the task for the NN. Data augmentation can easily be integrated by copying the image to random positions rather than aligning it to the left or by randomly resizing the image.

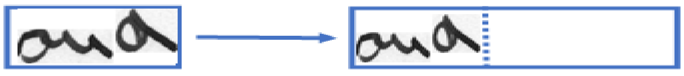
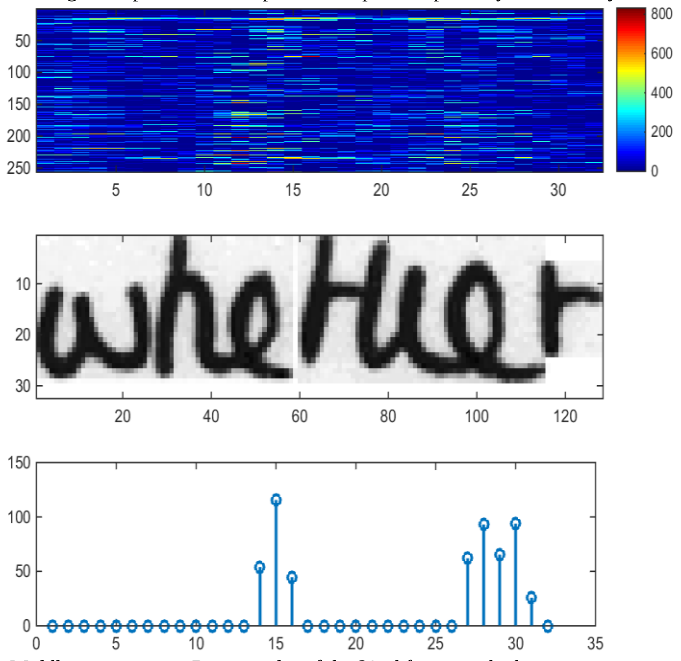


Fig. 4: Left: a picture from the dataset with an arbitrary size. it's scaled to suit the target image of size 128×32, the empty a part of the target image is crammed with white color.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

CNN output: Figure. 5 displays the output of the CNN layers which may be a sequence of length 32. Each layer entry contains 256 features. All these features are further process has been done by the RNN layers, however, some features already showed high correlation with certain high-level properties of the input image: there are some features which have a high correlation with characters for example "e", or with duplicate characters for example "ll", otherwise properties of character like loops as already present in handwritten “l”s or “e”s.

Fig. 5: Top: 256 features per time-step is computed by the CNN layers.



Middle: input image. Bottom: plot of the 32nd feature, which incorporates a high correlation with the occurrence of the character “e” within the image.

RNN output: Fig. 6 shows a visualization of the RNN output matrix for a picture containing the text “little”. The matrix shown within the top-most graph consists of the scores for the characters included in the Connectionist Temporal Classification blank label as its last entry. the opposite matrix-entries, from top to bottom, correspond to the subsequent characters: “ !”#&’()*+,-./ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnop qrstuvwxyz0123456789:;?”. Only the last character “e” isn't aligned. But this can be OK because the CTC operation is segmentation-free and doesn't care about absolute positions. From the bottom-most graph showing the scores for the characters “l”, “i”, “t”, “e” and also the CTC blank label, the text can easily be decoded: we just take the foremost probable character from each time-step, this forms the so-called best path, then we throw away repeated characters and at last all blanks: “l---ii--t-t--l-…-e” → “l---i--t-t--l-…-e” → “little”.

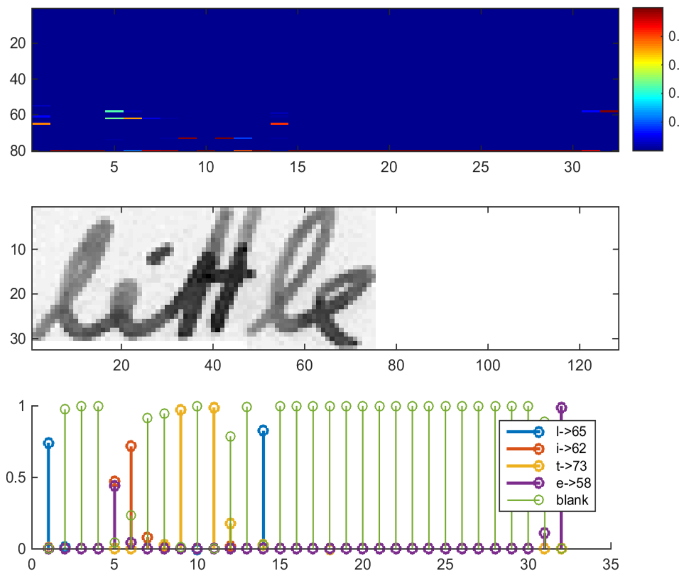


Fig. 6: Top: output matrix of the RNN layers. Middle: input image. Bottom: Probabilities for the characters “l”, “i”, “t”, “e” and therefore the CTC blank label.

D. *Implementation using TF:*

The implementation consists of 4 modules:

1. SamplePreprocessor.py: prepares the pictures from the IAM dataset for the NN
2. DataLoader.py: reads samples, put them into batches and provides an iterator-interface to travel through the info
3. Model.py: creates the model as described above, loads and saves models, manages the TF sessions and provides an interface for training and inference
4. main.py: puts all previously mentioned modules together

We only have a look at Model.py, because the other source files are concerned with basic file IO (DataLoader.py) and image processing (SamplePreprocessor.py).

CNN:

For each CNN layer, create a kernel of size k×k to be utilized in the convolution operation.

Then, RELU operation again to the pooling layer with size px×py and step-size sx×sy with results of the convolution.

These steps are repeated for all layers during a for-loop.

RNN:

Create and stack the two RNN layers consisting of 256 units each.

Then, create a bidirectional RNN from it, such the input sequence is traversed from front to back and therefore the other way round. As a result, we get two output sequences forward and backward of size 32×256, which we later concatenate along the feature-axis to create a sequence of size 32×512. Finally, it's mapped to the output sequence (or matrix) of size 32×80 which is fed into the CTC layer.

CTC:

For loss calculation, we feed both the bottom truth text and therefore the matrix to the operation. the bottom truth text is encoded as a sparse tensor. The length of the input sequences must be given to both CTC operations.

We now have all the input files to make the loss operation and therefore the decoding operation.

Training:

The mean of the loss values of the batch elements is employed to coach the NN: it's fed into an optimizer like RMSProp.

E. Improving the model:

In case you want to feed complete text-lines as shown in Fig. 6 instead of word-images, you have to increase the input size of the NN.

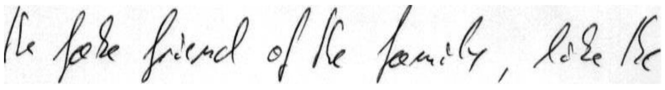

the fake friend of the family, like the

Fig. 7: A complete text-line can be fed into the NN if its input size is increased (image is taken from IAM).

If you want to improve the recognition accuracy, you can follow one of these hints:

- Data augmentation: increase dataset-size by applying further (random) transformations to the input images
- Remove cursive writing style in the input images
- Increase input size (if an input of NN is large enough, complete text-lines can be used)

F. Spell checker

Checking of spelling may be a basic requirement in any text processing or analysis. The python package **pyspellchecker** provides us this feature to search out the words that will are misspelled and also suggest the possible corrections.
First, we'd like to put in the specified package using the subsequent command in our python environment.

```
pip install pyspellchecker
```

III. THE ARCHITECTURE OF THE PROPOSED NETWORK

The proposed network has different layers which are as shown in the figure below:

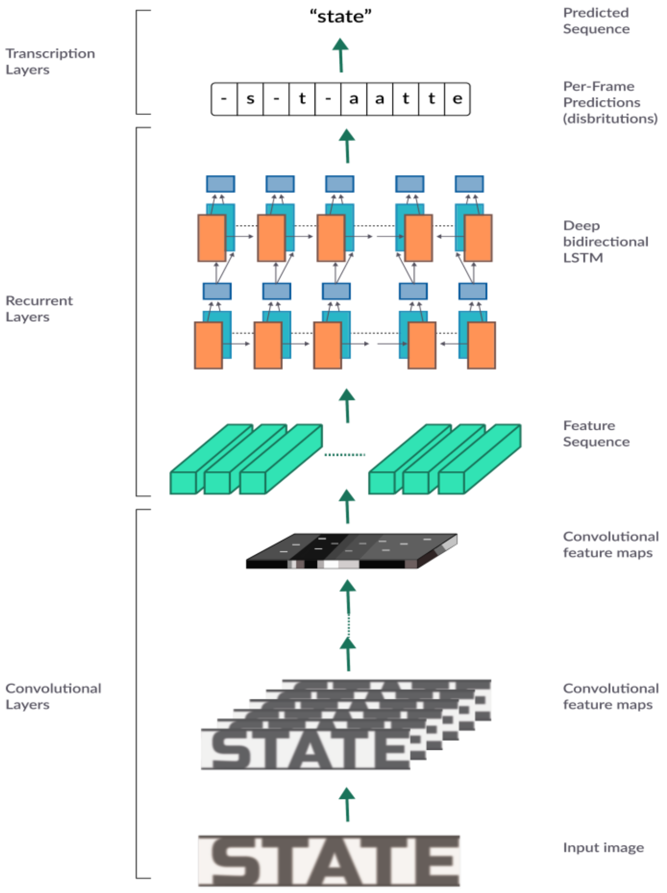


Fig. 8: Architecture of proposed network

A. CNN layers

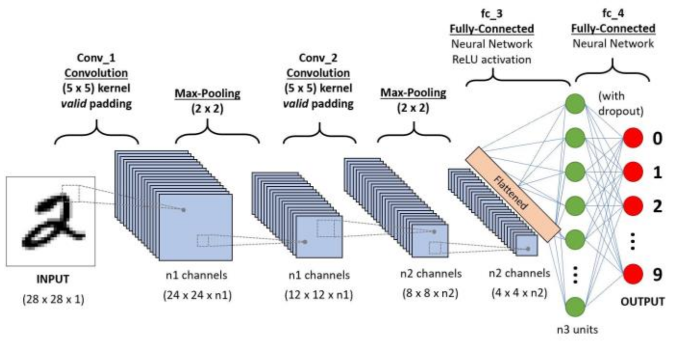
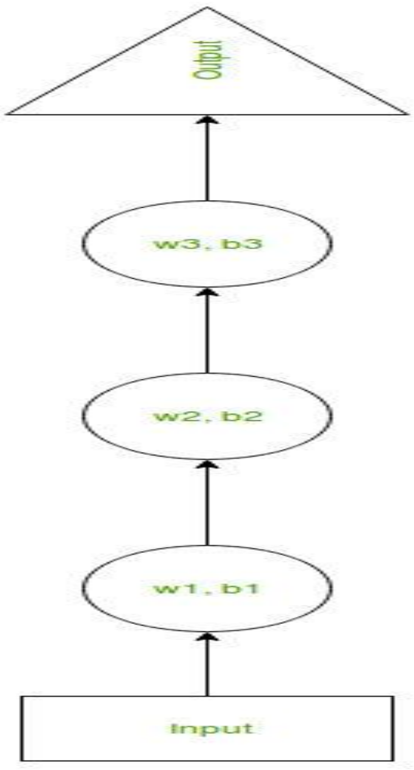

Conv_1 Convolution (5 x 5) kernel valid padding
Max-Pooling (2 x 2)
Conv_2 Convolution (5 x 5) kernel valid padding
Max-Pooling (2 x 2)
fc_3 Fully-Connected Neural Network ReLU activation
fc_4 Fully-Connected Neural Network
(with dropout)
Flattened
INPUT (28 x 28 x 1)
n1 channels (24 x 24 x n1)
n1 channels (12 x 12 x n1)
n2 channels (8 x 8 x n2)
n2 channels (4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT

Fig. 9: CNN layer

CNN is meant to imitate human visual processing, and it's highly optimized structures to process 2D images. Further, it can effectively learn the extraction and abstraction of 2D features. In detail, the max-pooling layer of CNN is extremely effective in absorbing shape variations. Moreover, a sparse reference to tied weights makes CNN involve with fewer parameters than a totally connected network with similar size. most significantly, CNN is trainable with the gradient-based learning algorithm and suffers less from the diminishing gradient problem. providing the gradient-based algorithm trains the entire network to attenuate a blunder criterion directly, CNN can produce highly optimized weights and good generalization performance.

B. RNN layer

RNN have a **"memory"** which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.



- RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous output by giving each output as input to the next hidden layer.
- Hence these three layers can be joined together such that the weights and bias of all the hidden layers are the same, into a single recurrent layer.
- *The formula for calculating current state:*

$$h_t = f(h_{t-1}, x_t)$$

- *The formula for applying Activation function(tanh):*

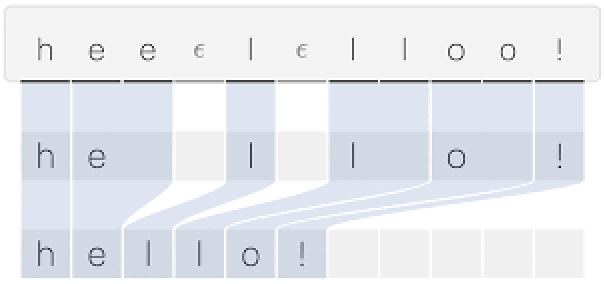
$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

- *Formula for calculating output:*

$$y_t = W_{hy}h_t$$

C. CTC layer

CTC may be a loss function that is employed to coach neural networks. there's no must align data because that may assign a probability for any label. CTC may be a align free.it works on the summing over the probability of all possible alignments between the input and also the label.
Here we have an input of size 9 and also the correct transmission of its iPhone. we force our system to assign an output character to every input step and so we collapse the repeats, which ends within the input.



Blank token:
There is how around that called the Blank token. It does mean anything and simply it gets removed before the final word output is produced. Now, the algorithm can assign a personality or blank token to each input step.

1. CTC network assigns the character with the simplest probability of every input sequence.
2. Repeats not having a blank token in between get merged.
3. Lastly, the blank token gets removed.

The CTC network can then give the probability label to the input. By summing all the probabilities of the characters of each time step.
The CTC algorithm is alignment-free — it doesn't require alignment between the input and thus the output. However, to induce the probability of output given an input, CTC works by summing over the probability of all possible alignments between the two. we'd wish to know what these alignments are so on know the way the loss function is ultimately calculated.
To motivate the precise sort of the CTC alignments, first consider a naive approach. Let's use an example. Assume that the given input has length six and Y =Y= [c, a, t]. a way to align XX and YY is to assign an output character to each input step and collapsed repeats. Experiment results
After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

D. Summary of Dataset

The IAM Handwriting Database contains forms of handwritten English text which can be used to train and test handwritten text recognizers and to perform writer identification and verification experiments.
Characteristics of IAM Dataset

- 657 writers contributed samples of their writings.
- 1532 pages of scanned text.
- 5685 isolated and labeled sentences.

- 13353 isolated and labeled text lines.
- 115320 isolated and labeled words.

Set name	Number of text lines	Number of writers
Train	6161	283
Validation 1	900	46
Validation 2	940	43
Test	1861	128
Total	9862	500

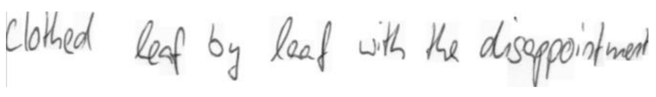
The above tables provide the information about one Training, one Testing, Two validation sets.

In the data set it contain many images of same types with a particular dimension and this data set also contain a label file with a text extension. It contains the image and its text. At first, it contains the image and followed by the particular text present in it.
The Labels.txt is a file which consists of data as follows:
IAM dataset line information
format: a01-000u-00 ok 154 19 408 746 1663 91 A|MOVE|to|stop|Mr.|Gaitskell|from
a01-000u-00 -> line id for form a01-000u
ok -> result of word segmentation
ok: line is correctly segmented
err: every part of a line has more than one error
notice: if the line cannot be properly segmented the transcription and extraction of the complete the line should not be affected negatively

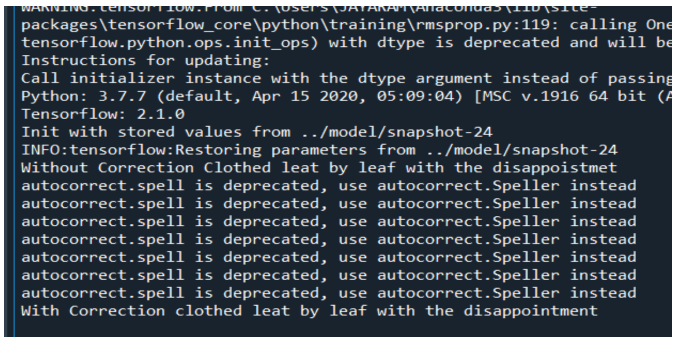
154 ->gray level to binarize line
19 -> number of components for this line
408 746 1663 91 -> bounding box through this line in x,y,w,h format
A|MOVE|to|stop|Mr.|Gaitskell|from -> transcription for this line. word tokens are separated by the character '|'

IV. RESULTS

In this project we have given image as an input then it predicts the output by loading the model which is already previously created and saved.



The above image is the input given to the neural network to predict the solution.



```
WARNING:tensorflow:From C:\Users\JAYARAM\Anaconda3\lib\site-
packages\tensorflow_core\python\training\rmsprop.py:119: calling One
tensorflow.python.ops.init_ops) with dtype is deprecated and will be
Instructions for updating:
Call initializer instance with the dtype argument instead of passing
Python: 3.7.7 (default, Apr 15 2020, 05:09:04) [MSC v.1916 64 bit (A
Tensorflow: 2.1.0
Init with stored values from ../model/snapshot-24
INFO:tensorflow:Restoring parameters from ../model/snapshot-24
Without Correction Clothed leat by leaf with the disappoistmet
autocorrect.spell is deprecated, use autocorrect.Speller instead
autocorrect.spell is deprecated, use autocorrect.Speller instead
autocorrect.spell is deprecated, use autocorrect.Speller instead
autocorrect.spell is deprecated, use autocorrect.Speller instead
autocorrect.spell is deprecated, use autocorrect.Speller instead
autocorrect.spell is deprecated, use autocorrect.Speller instead
autocorrect.spell is deprecated, use autocorrect.Speller instead
With Correction clothed leat by leaf with the disappointment
```

This is image which shows the output to the above input image.

V. CONCLUSION AND FUTURE SCOPE

In this project classification of characters takes place. The project is achieved through the conventional neural network. The accuracy we obtained in this is above 90.3%. This algorithm will provide both the efficiency and effective result for the recognition. The project gives best accuracy for the text which has less noise. The accuracy completely depending on the dataset if we increase the data, we can get more accuracy. If we try to avoid cursive writing then also its best results.

Future Work:

In future we are planning to extend this study to a larger extent where different embedding models can be considered on large variety of the datasets. The future is completely based on technology no one will use the paper and pen for writing. In that scenario they used write on touch pads so the inbuilt software which can automatically detects text which they writing and convert into digital text so that the searching and understanding very much simplified.

ACKNOWLEDGMENT

This work is partially supported by the grant from the Aditya Institute of Technology and Management. We thank M. Yugandhar for his guidance in this project.

REFERENCES

[1] Fischer, A., Frinken, V., Bunke, H.: Hidden Markov models for off-line cursive handwriting recognition, in C.R. Rao (ed.): Handbook of Statistics 31, 421 – 442, Elsevier, 2013

[2] Frinken, V., Bunke, H.: Continuous handwritten script recognition, in Doermann, D., Tombre, K. (eds.): Handbook of Document Image Processing and Recognition, Springer Verlag, 2014

[3] S. Günter and H. Bunke. A new combination scheme for HMM-based classifiers and its application to handwriting recognition. In Proc. 16th Int. Conf. on Pattern Recognition, volume 2, pages 332–337. IEEE, 2002.

[4] U.-V. Marti and H. Bunke. Text line segmentation and word recognition in a system for general writer independent handwriting recognition. In Proc. 6th Int. Conf. on Document Analysis and Recognition, pages 159–163.

[5] M. Liwicki and H. Bunke, "Iam-ondb - an on-line English sentence database acquired from the handwritten text on a whiteboard," in ICDAR, 2005

[6] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in Advances in neural information processing systems, 2009, pp. 545–552.

[7] Nafiz Arica, and Fatos T. Yarman-Vural, ―Optical Character Recognition for Cursive Handwriting,‖ IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.24, no.6, pp. 801-113, June 2002.

[8] Anita Pal and Davashankar Singh,"Handwritten English Character Recognition Using Neural Network", International Journal of Computer Science and Communication, pp: 141- 144, 2011.

[9] Sandhya Arora, "Combining Multiple Feature Extraction Techniques for Handwritten Devnagari Character Recognition", IEEE Region 10 Colloquium and the Third ICIIS, Kharagpur, INDIA, December 2008.

[10] Om Prakash Sharma, M. K. Ghose, Krishna Bikram Shah, "An Improved Zone Based Hybrid Feature Extraction Model for Handwritten Alphabets Recognition Using Euler Number", International Journal of Soft Computing and Engineering (ISSN: 2231 - 2307), Vol. 2, Issue 2, pp. 504- 508, May 2012.

[11] N. Venkata Rao and Dr. A.S.C.S.Sastry - ―Optical CharacterRecognition Technique Algorithms‖-2016 Journal of Theoretical and Applied Information Technology.

[12] J.Pradeep, E.Srinivasan and S.Himavathi ―Diagonal based feature extraction for handwritten alphabets recognition system using neural network‖ - International Journal of Computer Science & Information Technology (IJCSIT), Vol 3, No 1, Feb 2011.

[13] Manoj Sonkusare and Narendra Sahu "A SURVEY ON HANDWRITTEN CHARACTER RECOGNITION (HCR) TECHNIQUES FOR ENGLISH ALPHABETS" Advances in Vision Computing: An International Journal (AVC) Vol.3, No.1, March 2016 .