HANDWRITTEN CHARACTER RECOGNITION USING NEURAL NETWORKS

Artificial Intelligence Project Report

Semester : WINTER 2019-2020

FINAL REVIEW

SUBMITTED BY:-

D. SAHITH -> 17BCE0422

M.LIKITH CHOWDARY -> 18BIT0039

G.HRUSHIKESH MADHAV -> 18BCE0189

UNDER THE GUIDANCE OF

PEETHA EVANGELINE.D MADAM

SCOPE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Abstract---

Handwritten Text Recognition (HTR) system implemented with TensorFlow (TF) and trained on the IAM off-line HTR dataset. This Neural Network (NN) model recognizes the text contained in the images of segmented words we give as input. As these word-images are smaller than images of complete text-lines, the NN can be kept small and training on the CPU is feasible. 3/4 of the words from the validation-set are correctly recognized and the character error rate is around 10%.

Offline Handwritten Text Recognition (HTR) systems transcribe text contained in scanned images into digital text. We will build a Neural Network (NN) which is trained on word-images from the IAM dataset. As the input layer (and therefore also all the other layers) can be kept small for word-images, NN-training is feasible on the CPU (of course, a GPU would be better). This implementation is the bare minimum that is needed for HTR using TF.

Index Terms---

Character recognition, Data set, Handwritten text, Image segmentation, Neural Networks, Probability, Tenserflow

I. INTRODUCTION

Character recognition is a craft of recognizing, dividing and distinguishing characters from image. An extreme goal of written by hand character recognition is to reproduce the human perusing capacities so the PC can peruse comprehend alter and function as human do with text. Handwriting recognition has been a standout amongst the most entrancing and testing research zones in field of image processing and pattern recognition in the ongoing years. It contributes colossally to the progression of mechanization process and improves the interface among man and machine in various applications. A few research works have been concentrating on new strategies and techniques that would diminish the handling time while giving higher recognition accuracy. Character recognition is predominantly of two kinds on the web and off-line. The purpose of this project is to take handwritten English characters as input, process the characters, train the neural network algorithm, to recognize the pattern and modify the character to a beautified version of the input. This project is aimed at developing software which will be helpful in recognizing Characters of English language. This project is restricted to English characters only.it can be further developed to recognize the characters of different languages. It engulfs the concept of neural network. One of the primary means by which computers are endowed with humanlike abilities is through the use of neural network. Neural networks are particularly useful for solving problems that cannot be expressed as a series of steps, such as recognizing patterns, classifying them into groups, series prediction and data mining. Pattern recognition is perhaps the most common use of neural networks. The neural network is presented with a target vector and also a vector which contains the pattern information, this could be an image and hand written data. The neural network then attempts to determine if the input data matches a pattern that the neural network has memorized. A neural network trained for classification is designed to take input samples and classify them into groups. These groups may be fuzzy, without clearly defined boundaries. This project concerns detecting free handwritten characters.

II. LITERATURE SURVEY

1.Handwritten Character Recognition using Neural Network

Goal of this paper is to recognize the characters in a given filtered records and concentrate the impacts of changing the Models of ANN. Today Neural Networks are for the most part utilized for Pattern Recognition task. The paper portrays the practices of various Models of Neural Network utilized in OCR. OCR is boundless utilization of Neural Network. We have considered parameters like number of Hidden Layer, size of Hidden Layer and ages. We have utilized Multilayer Feed Forward system with Back spread. In Pre-processing we have connected some fundamental calculations for division of characters, normalizing of characters and De-skewing. We have utilized diverse Models of Neural Network and connected the test set on each to discover the precision of the individual Neural Network.

2. Handwritten Character Recognition Using Gradient Features

Feature extraction is a fundamental piece of any recognition framework. The point of Feature extraction is to depict the example by methods for least number of features that are powerful in segregating pattern classes. The gradient estimates the greatness and heading of the greatest change in power in a little neighbourhood of each pixel. (In what pursues, "Gradient" alludes to both the gradient size and direct particle). Inclinations are computed by methods for the Sobel operator. In this paper an exertion is made towards recognition of English Characters and acquired recognition precision of 94%. Due to its consistent straightforwardness, usability and high recognition rate, Gradient Features ought to be utilized for recognition purposes.

3. A Review of Gradient-Based and Edge-Based Feature Extraction Methods for Object Detection.

In PC vision look into, object location dependent on image processing is the assignment of distinguishing an assigned article on a static image or a grouping of video outlines. Ventures dependent on such research works have been broadly adjusted to different modern and social applications. The field to which those applications apply incorporates however not constrained to, security reconnaissance, clever transportation n system, computerized assembling, and quality control and inventory network the executives. In this paper, we are going to survey a couple of most well known PC vision techniques dependent on image processing and example recognition. Those techniques have been broadly contemplated in different research papers and their noteworthiness to PC vision look into has been demonstrated by ensuing examination works. When all is said in done, we arrange those techniques into to inclination based and edge based element extraction strategies, contingent upon the low dimension highlights they use. In this paper, the definitions for gradient and edge are expanded. Since a image can likewise be considered as a lattice of image patches, it is in this manner sensible to fuse the idea of granules to inclination for an audit.

III. METHODOLOGY

We use a NN for our task. It consists of convolutional NN (CNN) layers, recurrent NN (RNN) layers and a final Connectionist Temporal Classification (CTC) layer. Fig. 2 shows an overview of our HTR system.

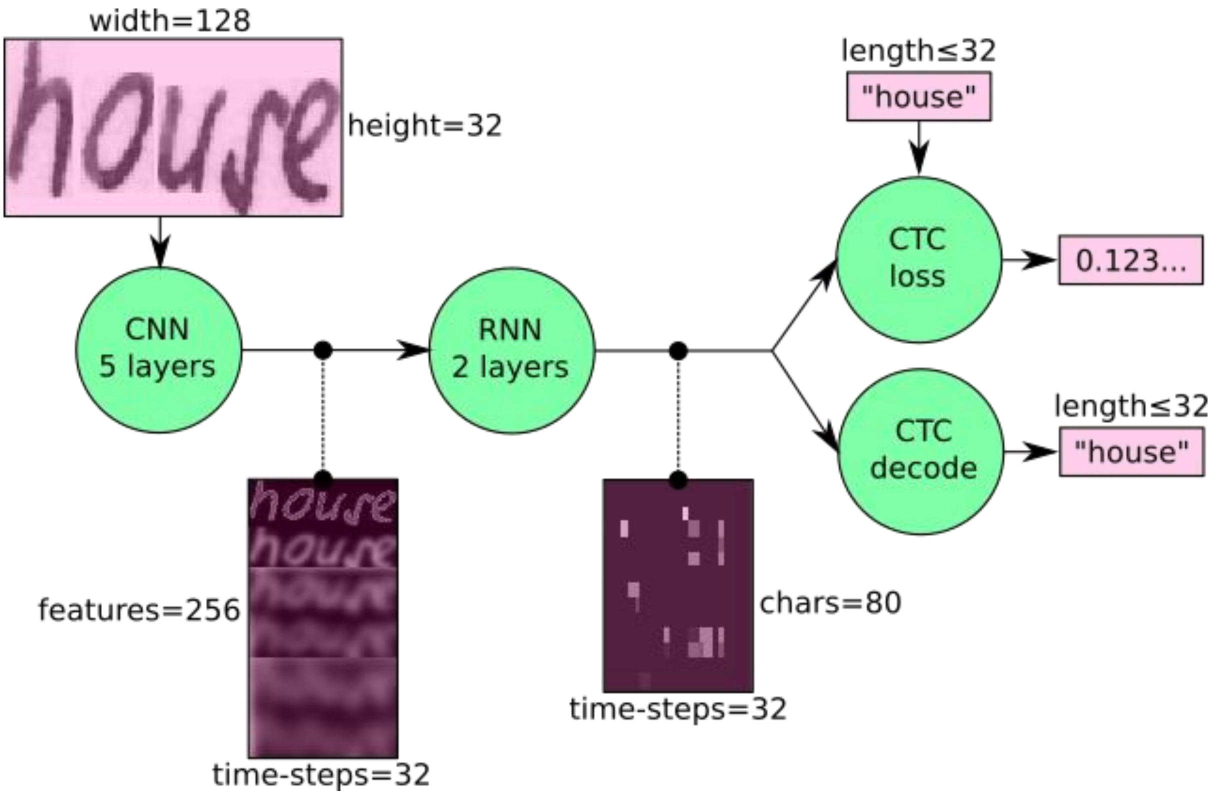


Fig. 2: Overview of the NN operations (green) and the data flow through the NN (pink).

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

$$\text{NN}: \underset{W\times H}{M} \rightarrow \underset{0\le n\le L}{(c_1, c_2, \ldots, c_n)}$$

Eq. 1: The NN written as a mathematical function which maps an image M to a character sequence (c1, c2, …).

Operations

CNN: the input image is fed into the CNN layers. These layers are trained to extract relevant features from the image. Each layer consists of three operation. First, the convolution operation, which applies a filter kernel of size 5×5 in the first two layers and 3×3 in the last three layers to the input. Then, the non-linear RELU function is applied. Finally, a pooling layer summarizes image regions and outputs a downsized version of the input. While the image height is downsized by 2 in each layer, feature maps (channels) are added, so that the output feature map (or sequence) has a size of 32×256.

RNN: the feature sequence contains 256 features per time-step, the RNN propagates relevant information through this sequence. The popular Long Short-Term Memory (LSTM) implementation of RNNs is used, as it is able 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 size 32×80. The IAM dataset consists of 79 different characters, further one additional character is needed for the CTC operation (CTC blank label), therefore there are 80 entries for each of the 32 time-steps.

CTC: while training the NN, the CTC is given the RNN output matrix and the ground truth text and it computes the **loss value**. While inferring, the CTC is only given the matrix and it decodes it into the **final text**. Both the ground truth text and the recognized text can be at most 32 characters long.

Data

Input: it is a gray-value image of size 128×32. Usually, the images from the dataset do not have exactly this size, therefore we resize it (without distortion) until it either has a width of 128 or a height of 32. Then, we copy the image into a (white) target image of size 128×32. This process is shown in Fig. 3. Finally, we normalize the gray-values of the image which simplifies the task for the NN. Data augmentation can easily be integrated by copying the image to random positions instead of aligning it to the left or by randomly resizing the image.

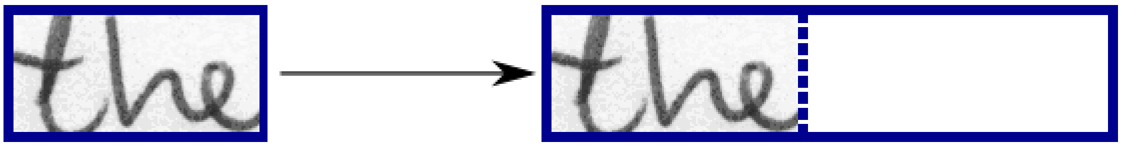


Fig. 3: Left: an image from the dataset with an arbitrary size. It is scaled to fit the target image of size 128×32, the empty part of the target image is filled with white color.

CNN output: Fig. 4 shows the output of the CNN layers which is a sequence of length 32. Each entry contains 256 features. Of course, these features are further processed by the RNN layers, however, some features already show a high correlation with certain high-level properties of the input image: there are features which have a high correlation with characters (e.g. "e"), or with duplicate characters (e.g. "tt"), or with character-properties such as loops (as contained in handwritten "l"s or "e"s).

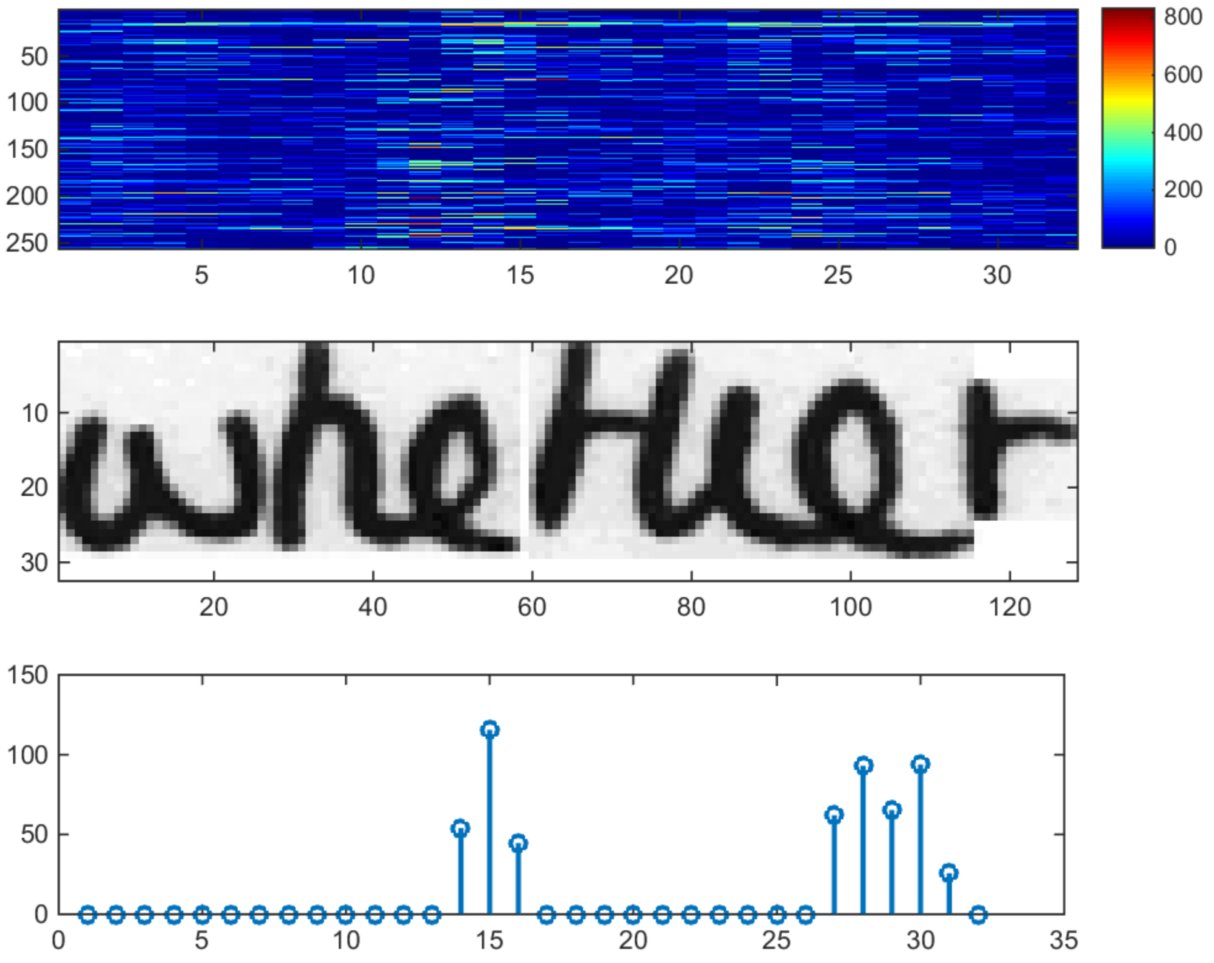


Fig. 4: Top: 256 feature per time-step are computed by the CNN layers. Middle: input image. Bottom: plot of the 32nd feature, which has a high correlation with the occurrence of the character "e" in the image.

RNN output: Fig. 5 shows a visualization of the RNN output matrix for an image containing the text "little". The matrix shown in the top-most graph contains the scores for the characters including the CTC blank label as its last (80th) entry. The other matrix-entries, from top to bottom, correspond to the following characters: " !"#&'()*+,-./0123456789:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz". It can be seen that most of the time, the characters are predicted exactly at the position they appear in the image (e.g. compare the position of the “i” in the image and in the graph). Only the last character “e” is not aligned. But this is OK, as the CTC operation is segmentation-free and does not care about absolute positions. From the bottom-most graph showing the scores for the characters “l”, “i”, “t”, “e” and the CTC blank label, the text can easily be decoded: we just take the most probable character from each time-step, this forms the so called best path, then we throw away repeated characters and finally all blanks: “l---ii--t-t--l-...-e” → “l---i--t-t--l-...-e” → “little”.

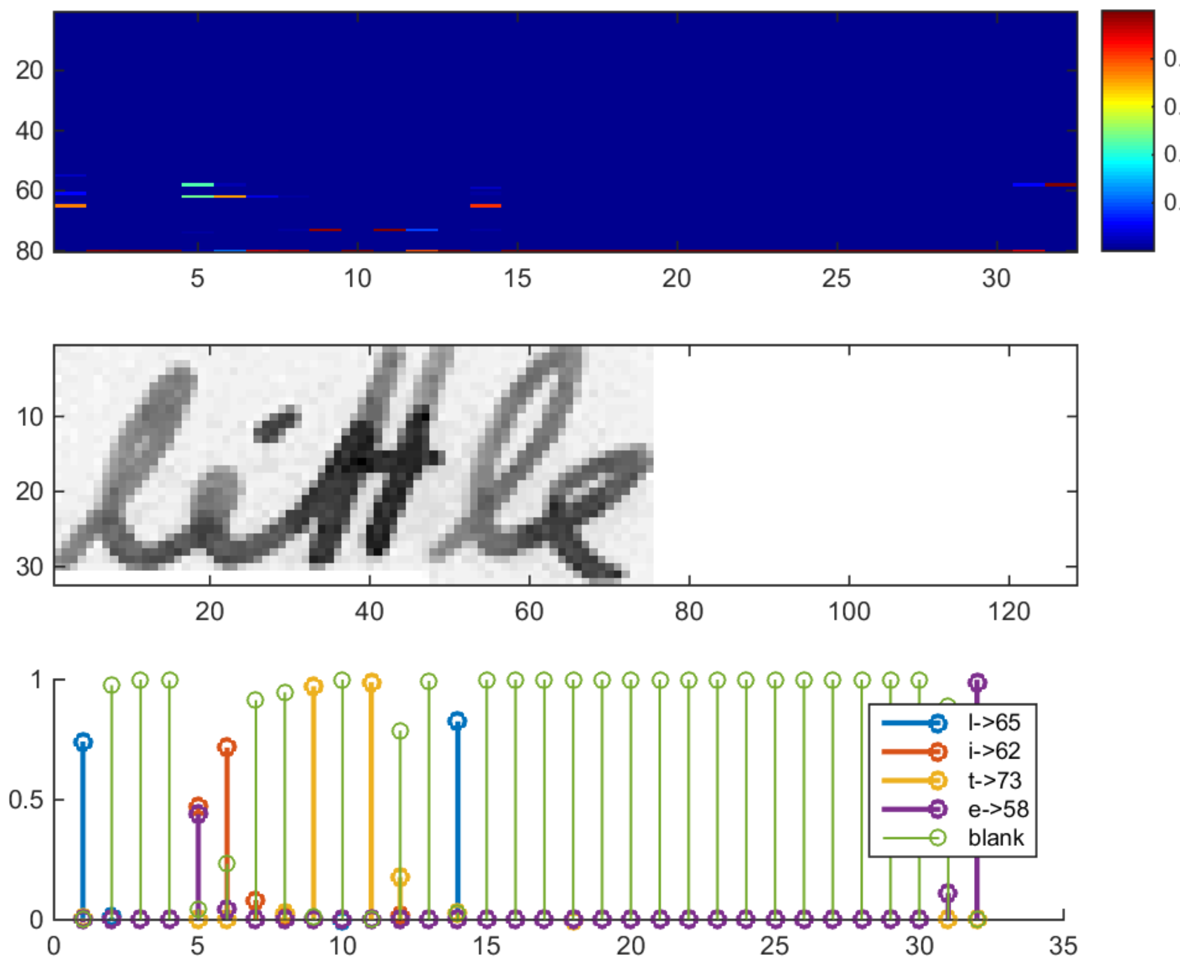


Fig. 5: Top: output matrix of the RNN layers. Middle: input image. Bottom: Probabilities for the characters “l”, “i”, “t”, “e” and the CTC blank label.

Implementation using TF

The implementation consists of 4 modules:

1. SamplePreprocessor.py: prepares the images from the IAM dataset for the NN
2. DataLoader.py: reads samples, puts them into batches and provides an iterator-interface to go through the data

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 look at Model.py, as 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 used in the convolution operation.

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

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

RNN

Create and stack two RNN layers with 256 units each.

Then, create a bidirectional RNN from it, such that the input sequence is traversed from front to back and the other way round. As a result, we get two output sequences fw and bw of size 32×256, which we later concatenate along the feature-axis to form a sequence of size 32×512. Finally, it is 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 ground truth text and the matrix to the operation. The ground truth text is encoded as a sparse tensor. The length of the input sequences must be passed to both CTC operations.

We now have all the input data to create the loss operation and the decoding operation.

Training

The mean of the loss values of the batch elements is used to train the NN: it is fed into an optimizer such as RMSProp.

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.

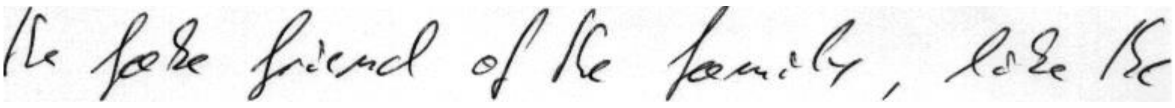
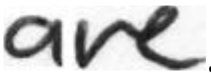


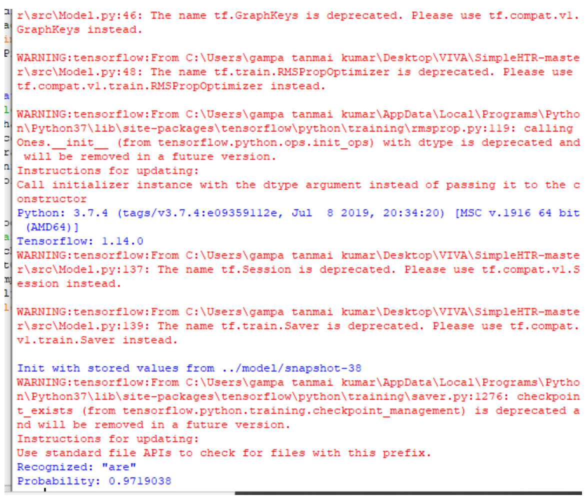
Fig. 6: A complete text-line can be fed into the NN if its input size is increased (image taken from IAM).

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

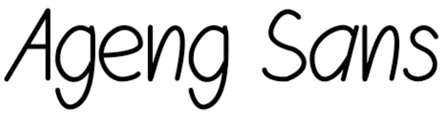
- Data augmentation: increase dataset-size by applying further (random) transformations to the input images
- Remove cursive writing style in the input images (see DeslantImg)
- Increase input size (if input of NN is large enough, complete text-lines can be used)
- Add more CNN layers
- Replace LSTM by 2D-LSTM
- Decoder: use token passing or word beam search decoding (see CTCWordBeamSearch) to constrain the output to dictionary words
- Text correction: if the recognized word is not contained in a dictionary, search for the most similar one

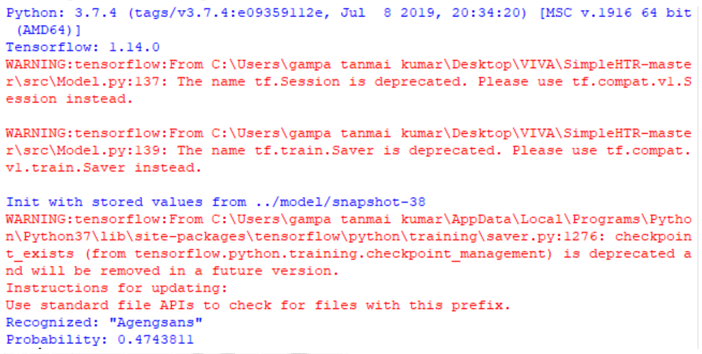
IV. EXPERIMENTATIONS AND RESULTS


.

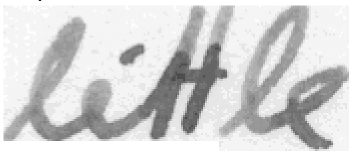


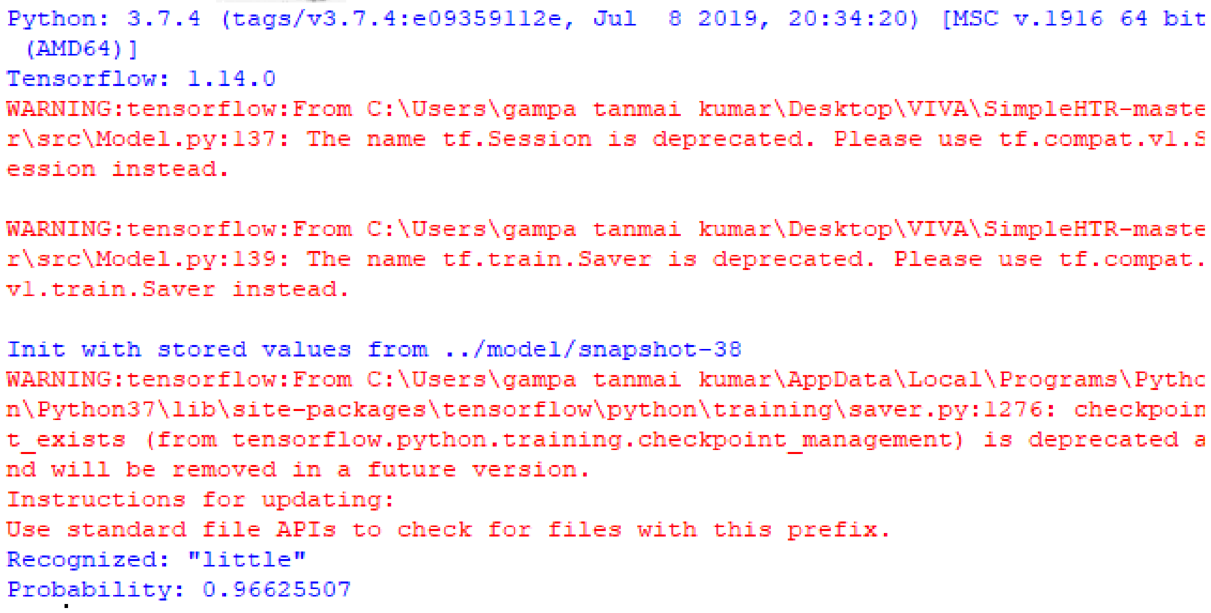
```
r\src\Model.py:46: The name tf.GraphKeys is deprecated. Please use tf.compat.v1.
GraphKeys instead.

WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:48: The name tf.train.RMSPropOptimizer is deprecated. Please use
tf.compat.v1.train.RMSPropOptimizer instead.

WARNING:tensorflow:From C:\Users\gampa tanmai kumar\AppData\Local\Programs\Pytho
n\Python37\lib\site-packages\tensorflow\python\training\rmsprop.py:119: calling
Ones.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and
 will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the c
onstructor
Python: 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
 (AMD64)]
Tensorflow: 1.14.0
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:137: The name tf.Session is deprecated. Please use tf.compat.v1.S
ession instead.

WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:139: The name tf.train.Saver is deprecated. Please use tf.compat.
v1.train.Saver instead.

Init with stored values from ../model/snapshot-38
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\AppData\Local\Programs\Pytho
n\Python37\lib\site-packages\tensorflow\python\training\saver.py:1276: checkpoin
t_exists (from tensorflow.python.training.checkpoint_management) is deprecated a
nd will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
Recognized: "are"
Probability: 0.9719038
```



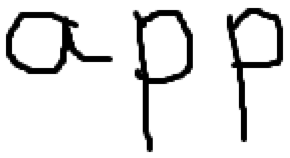


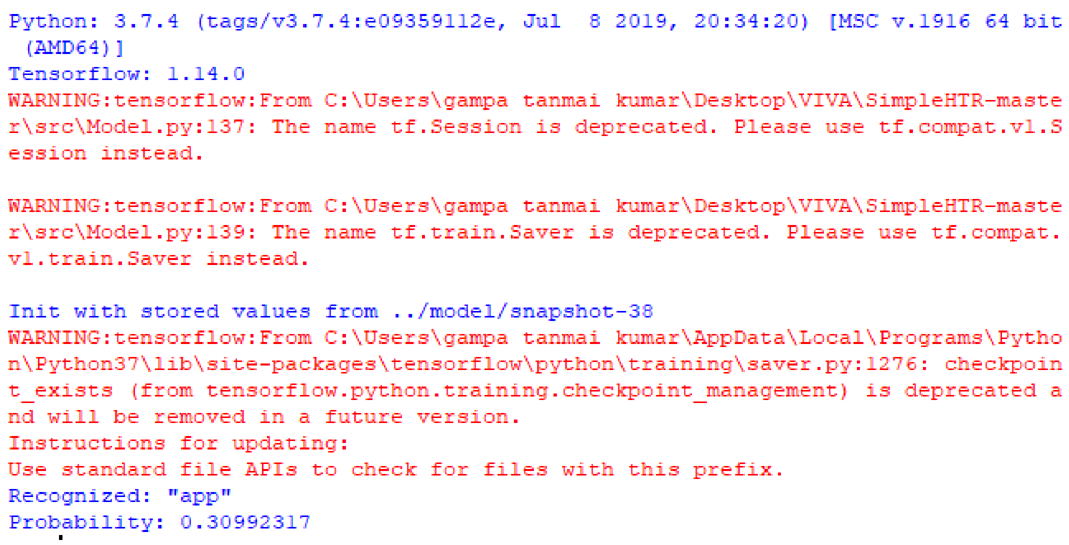
```
Python: 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
 (AMD64)]
Tensorflow: 1.14.0
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:137: The name tf.Session is deprecated. Please use tf.compat.v1.S
ession instead.

WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:139: The name tf.train.Saver is deprecated. Please use tf.compat.
v1.train.Saver instead.

Init with stored values from ../model/snapshot-38
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\AppData\Local\Programs\Pytho
n\Python37\lib\site-packages\tensorflow\python\training\saver.py:1276: checkpoin
t_exists (from tensorflow.python.training.checkpoint_management) is deprecated a
nd will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
Recognized: "Agengsans"
Probability: 0.4743811
```



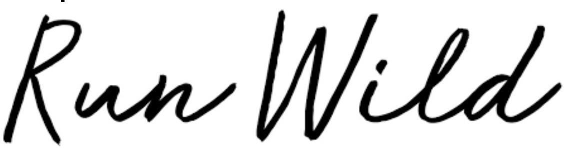


```
Python: 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
 (AMD64)]
Tensorflow: 1.14.0
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:137: The name tf.Session is deprecated. Please use tf.compat.v1.S
ession instead.

WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:139: The name tf.train.Saver is deprecated. Please use tf.compat.
v1.train.Saver instead.

Init with stored values from ../model/snapshot-38
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\AppData\Local\Programs\Pyth
n\Python37\lib\site-packages\tensorflow\python\training\saver.py:1276: checkpoin
t_exists (from tensorflow.python.training.checkpoint_management) is deprecated a
nd will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
Recognized: "little"
Probability: 0.96625507
```

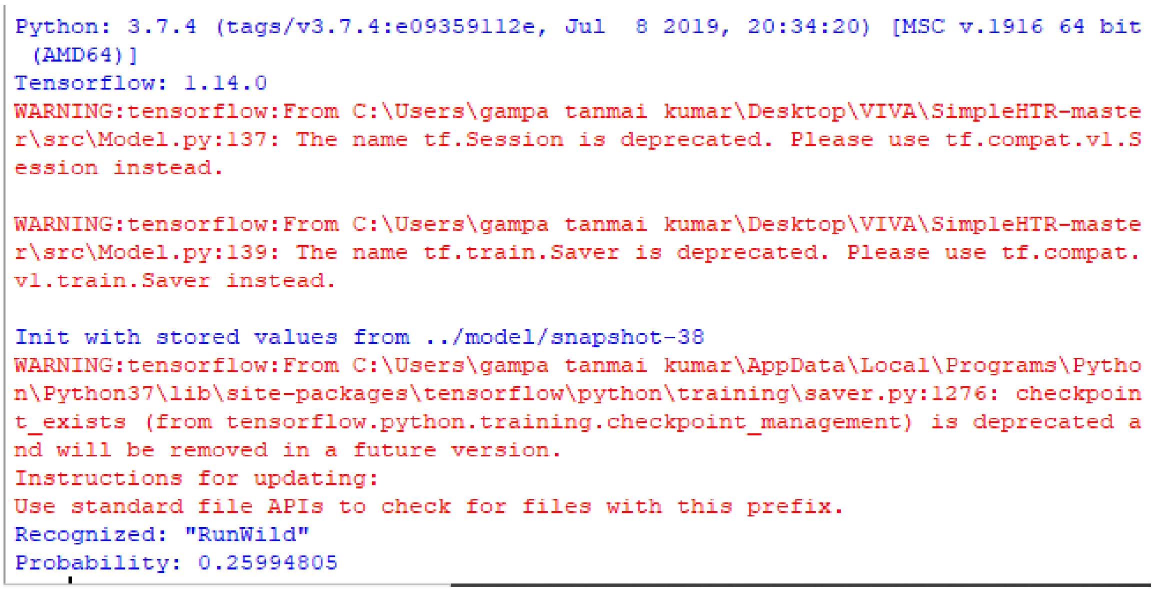




```
Python: 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
 (AMD64)]
Tensorflow: 1.14.0
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:137: The name tf.Session is deprecated. Please use tf.compat.v1.S
ession instead.

WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:139: The name tf.train.Saver is deprecated. Please use tf.compat.
v1.train.Saver instead.

Init with stored values from ../model/snapshot-38
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\AppData\Local\Programs\Pytho
n\Python37\lib\site-packages\tensorflow\python\training\saver.py:1276: checkpoin
t_exists (from tensorflow.python.training.checkpoint_management) is deprecated a
nd will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
Recognized: "app"
Probability: 0.30992317
```



Run Wild



```
Python: 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
 (AMD64)]
Tensorflow: 1.14.0
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:137: The name tf.Session is deprecated. Please use tf.compat.v1.S
ession instead.

WARNING:tensorflow:From C:\Users\gampa tanmai kumar\Desktop\VIVA\SimpleHTR-maste
r\src\Model.py:139: The name tf.train.Saver is deprecated. Please use tf.compat.
v1.train.Saver instead.

Init with stored values from ../model/snapshot-38
WARNING:tensorflow:From C:\Users\gampa tanmai kumar\AppData\Local\Programs\Pytho
n\Python37\lib\site-packages\tensorflow\python\training\saver.py:1276: checkpoin
t_exists (from tensorflow.python.training.checkpoint_management) is deprecated a
nd will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
Recognized: "RunWild"
Probability: 0.25994805
```

V. CONCLUSIONS

The feature extraction methods have performed well in classification when fed to the neural network and pre-processing of image using edge detection and normalization are the ideal choice for degraded noisy images. The method of training neural network with extracted features from sample images of each character has detection accuracy to a greater extent. The proposed methodology has produced good results for images containing handwritten text written in different styles, different size and alignment with varying background. We discussed a Neural Network which is able to recognize text in images. The Neural

Network consists of 5 CNN and 2 RNN layers and outputs a character-probability matrix. This matrix is either used for CTC loss calculation or for CTC decoding. An implementation using TF is provided and some important parts of the code were presented. Finally, hints to improve the recognition accuracy were given.

REFERENCES :-

1. W. L. Goh, D. P. Mital and H. A. Babri, "An artificial neural network approach to handwriting recognition," Proceedings of 1st International Conference on Conventional and Knowledge Based Intelligent Electronic Systems. KES '97, Adelaide, SA, Australia, 1997, pp. 132-136 vol.1.
2. Matthew Ziegler, “Handwritten Numeral Recognition via Neural Networks with Novel Preprocessing Schemes”.
3. Li, Zhe et al. “Rotation-Free Online Handwritten Chinese Character Recognition using Two-Stage Convolutional Neural Network.” 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR) (2018): 205-210.
4. Nautiyal C.T., Singh S., Rana U.S. (2018) Recognition of Noisy Numbers Using Neural Network. In: Pant M., Ray K., Sharma T., Rawat S., Bandyopadhyay A. (eds) Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing, vol 584. Springer, Singapore
5. Y. Jason C. Jeff B. Yoshua L. Hod How Transferable are Features in Deep neural networks 2012 [online] Available: http://papers.nips.cc/paper/5347-how-transferable-arefeatures-in-deep-neural-networks.pdf
6. K. Gopalakrishnan S.K. Khaitan Alok Choudhary Ankit Agrawal "Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection" Construction and Building Materials vol. 157 pp. 322-330 2017.
7. S. Alam R. Tahsin D. Mohammad Rashed Humayun Ahmed NumtaDB-Assembled Bengali Handwritten Digits 2018.
8. B. Mithun K. Gautam Shom I. Rafiqul Md. Shopon M. Nabeel M. Sifat A. Anowarul "BanglaLekha-Isolated: A multi-purpose comprehensive dataset of Handwritten Bangla Isolated characters" Data in Brief 2017.
9. S. M. A. Sharif N. Mohammed N. Mansoor S. Momen "A hybrid deep model with HOG features for Bangla handwritten numeral classification" 2016 9th International Conference on pp. 463-466 2016 December.
10. M. Rusiñol D. Aldavert R. Toledo J. Lladós "Towards Query-by-Speech Handwritten Keyword Spotting" ICDAR pp. 501-505 2015.
11. L. Rothacker G. A. Fink "Segmentation-free Query-by-String Word Spotting with Bag-of Features HMMs" ICDAR Nancy France pp. 661-665 2015.
12. Ashiquzzaman, Akm & Tushar, Abdul Kawsar. (2017). Handwritten Arabic Numeral Recognition using Deep Learning Neural Networks. 10.1109/ICIVPR.2017.7890866.