Handwritten text identification Using Deep Learning in Python

A REPORT

Submitted by

K PAVAN KARTHIK 17MIS1038

In partial fulfilment for the award

Of

**M. Tech. Software Engineering
(5 year Integrated Programme)**

School of Computing Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

September 2020

VIT®

Vellore Institute of Technology

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

School of Computing Science and Engineering

DECLARATION

I hereby declare that the project entitled **“Hand Written Text Recognition Using Deep Learning in Python”** submitted by me to the School of Computing Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai 600127 in partial fulfilment of the requirements for the award of the degree of **Master of Technology -Software Engineering (Integrated)** is a record of bonafide work carried out by me**.** I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature

K Pavan Karthik
(17MIS1038)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

CERTIFICATE

The project report entitled "Hand **Written Text recognition using deep learning**" is prepared and submitted by "**K Pavan Karthik (17MIS1038)**". It has been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements for the award of the degree of **Master of Technology – Software Engineering (Integrated)** in Vellore Institute of Technology, Chennai, India.

Examined by:

Examiner I
Examiner II

PADAAYI

www.padaayi.blogspot.in
8331926163(India)
Making education affordable

UAM :- TN02D0038536

19/06/2020

To Whom so ever it May Concern

வணக்கம்/ నమస్కారము / नमस्कार / ನಮಸ್ಕಾರ/ নমস্কার/ નમસ્તે/ ସୁପ୍ରଭାତ/ നമസ്കാരം

This is to certify that **Pavan Karthik (పవన్ కార్తిక్ பவன் கார்டிக்),** a student of M.Tech Integrated Software Engineering of VIT University, Chennai Campus,Tamil Nadu has successfully completed internship program at **PaDaayi**. from(May 1, 2020 to May 30 , 2020). During this period, he has worked on **"Handwritten text identification for the 'Bharati script' Using Deep Learning in Python".**

It has been noticed that he is friendly and has special interest in doing projects that will impact the society positively.

We wish to continue association with such soft natured boy and wish to see him coming out with more projects(/solutions) that are relevant for the society

पि. विक्रम कुमार
పి. విక్రమ్ కుమార్
P. vikram Kumar (8331926163 always BSNL)
Science/Maths promoter
Founder Member(Padaayi)

* For details about the script: www.bharatiscript.com

శాకాహారమే తినవలె సత్యము పలుకవలె

Plot 15	opp Dr. ShEshAchAry's house	Ganesh Nagar colony	Marredpally	Secunderabad	500026	Andhra Pradesh

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to the HOD 5 Years Integrated M. Tech Software Engineering **(Dr. Asnath Victy Phamila Y)** for her exemplary guidance and encouragement for the internship. The blessing and guidance given by Dean of SCSE, VIT Chennai (**Dr. Jagadeesh Kannan R)** time to time shall carry me a long way in the journey of life on which I am about to embark.

I also take this opportunity to express a deep sense of gratitude to the Associate Dean of SCSE, VIT Chennai (**Dr. Geetha S)** for her cordial support, valuable information and guidance. I also like to thank the faculty coordinator **(Dr. A. Menaka Pushpa mam)** for spending her valuable time to guide and monitor the work carried regularly.

The internship opportunity I had with **"PADAAYI"** was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am obliged to staff for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my internship.

Lastly, I thank almighty, my parents, brother and friends for their constant encouragement without which this assignment would not be possible. I would like to express my deepest appreciation to all those who provided me the possibility to complete this internship.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Expansion
CNN	Convolution Neural Network
HTR	HandWritten Text Recognition
RNN	Recurrent Neural Network
CTC	Connectionist Temporal Classification
NN	Neural Network
CPU	Central Processing Unit
LSTM	Long Short-Term Memory
RELU	Peak Expiratory Flow Rate
GPU	Graphical Processing Unit

ABSTRACT

The project on which I worked was a Hand written Text Recognition. Here we will be recognizing the Telugu hand written characters using Deep Learning. There are 56 Telugu Characters by Developing the model to train all the hand written characters and recognize the characters. The characters are hand written and scanned with a camera and used for training the model. The present scenario is having the text recognition for English and many foreign languages but we are having difficulty for Indian languages. What is the main problem addressed is we are having many troubles to identify the Indian languages Because India is having many languages when compared to other countries, the difficult part is that we are having all type of characters in similar manner to develop the model we need to have more accuracy and Less error rate, There are many languages which doesn’t have the Hand text recognition Including Mother Language (TELUGU) Due to many reasons. The main reason is no sufficient data is there for training the model and predicting the trained data and also the language priority is also a another reason in INDIA 60 % of the people are speaking Hindi so they had given the main priority to the Hindi and they had Neglected many other languages,So we took our mother language to recognize hand written text

The present scenario on which the team is working on the various Indian languages and we took Telugu, so our Contribution In the project is we are trying to develop the model to recognize Telugu hand written characters. To have the boundry of the text area present in the Image we had also developed the Bounding box to have the text boundaries where we can draw the boundry of the text and it will capture the text and write it into a file name, the file name will be the Image file name. By capturing the boundaries of all the characters, we are training the model by the end of boundaries will be the character present in that boundry and then it will be feed into the CNN. We collected all the data by ourselves to train the model by various sources. We had developed the CNN model for feature extraction and then the output of the CNN model is fed into the RNN model. And after that we hand went for CTC Operation to finding the loss

1. INTRODUCTION

TENSORFLOW

TensorFlow offers different degrees of reflection so you can pick the correct one for your necessities. Assemble and train models by utilizing the elevated level Keras API, which makes beginning with TensorFlow and AI simple.

In the event that you need greater adaptability, excited execution takes into consideration prompt emphasis and natural investigating. For enormous ML preparing assignments, utilize the Distribution Strategy API for circulated preparing on various equipment designs without changing the model definition.

TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++.

TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

FEATURES OF THE LIBRARY

- Responsive Construct With TensorFlow we can easily visualize each and every part of the graph which is not an option while using Numpy or SciKit.
- Flexible One of the very important Tensorflow Features is that it is flexible in its operability, meaning it has modularity and the parts of it which you want to make standalone, it offers you that option.
  Do you know about TensorFlow Linear Model
- Easily Trainable It is easily trainable on CPU as well as GPU for distributed computing.

- Parallel Neural Network Training TensorFlow offers pipelining in the sense that you can train multiple neural networks and multiple GPUs which makes the models very efficient on large-scale systems.
- Mathematics with TensorFlow Tensors are the basic data structures in TensorFlow, and they represent the connecting edges in a dataflow graph.
- Defining Computational Graphs The good thing about working with dataflow graphs is that the execution model is separated from its execution (on CPU, GPU, or some combination) where, once implemented, software in TensorFlow can be used on the CPU or GPU where all complexity related to code execution is hidden.
- Simple Expressions Before we move on to discuss elements of TensorFlow, we will first do a session of working with TensorFlow, to get a feeling of what a TensorFlow program looks like.
- Sessions In order to actually evaluate the nodes, we must run a computational graph within a session.
- Matrix Operations Matrix operations are very important for machine learning models, like linear regression, as they are often used in them. TensorFlow supports all the most common matrix operations, like multiplication, transposing, inversion, calculating the determinant, solving linear equations, and many more.
-

- **NUMPY IN PYTHON**

NumPy is an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines. Therefore, the library contains a large number of mathematical, algebraic, and transformation functions. NumPy is an extension of Numeric and Numarray. NumPy also contains random number generators. NumPy is a wrapper around a library implemented in C. Pandas objects rely heavily on NumPy objects. Essentially, Pandas extends Numpy.

  - One Dimensional ArrayOne of the most important objects is an N-dimensional array type known as ndarray. We can think of a one-dimensional array as a column or a row of a table with one or more elements:
  - Multi-Dimensional Array A multidimensional array has more than one column

- **KERAS APPLICATION**

Keras is a high-level library that's built on top of Theano or TensorFlow. It provides a scikit-learn type API (written in Python) for building Neural Networks. Developers can use Keras to quickly build neural networks without worrying about the mathematical aspects of tensor algebra, numerical techniques, and optimization methods. The key idea behind the development of Keras is to facilitate experimentations by fast prototyping. The ability to go from an idea to result with the least possible delay is key to good research.

❖ **FEATURES OF THE LIBRARY**

- Keras is a high-level interface and uses Theano or Tensorflow for its backend.
- It runs smoothly on both CPU and GPU.
- Keras supports almost all the models of a neural network – fully connected, convolutional, pooling, recurrent, embedding, etc.
- Keras Sequential Models As a review, Keras provides a Sequential model API.
- Keras Functional Models Convolutional Neural Network
- In this section, we will define a convolutional neural network for image classification.
- The Keras functional API provides a more flexible way for defining models.
- Convolutional Neural Network In this section, we will define a convolutional neural network for image classification.

- **TENSORBOARD**

Tensorboard is the interface used to visualize the graph and other tools to understand, debug, and optimize the model.

Scalars: Show different useful information during the model training

Graphs: Show the model

Histogram: Display weights with a histogram

Distribution: Display the distribution of the weight

Projector: Show Principal component analysis and T-SNE algorithm. The technique uses for dimensionality reduction

A neural network decides how to connect the different "neurons" and how many layers before the model can predict an outcome.

Once you have defined the architecture, you not only need to train the model but also a metrics to compute the accuracy of the prediction. This metric is referred to as a **loss function.**

The objective is to minimize the loss function. In different words, it means the model is making fewer errors. All machine learning algorithms will repeat many times the computations until the loss reach a flatter line. To minimize this loss function, you need to define a **learning rate.**

It is the speed you want the model to learn. If you set a learning rate too high, the model does not have time to learn anything. This is the case in the left picture. The line is moving up and down, meaning the model predicts with pure guess the outcome. The picture on the right shows that the loss is decreasing over iteration until the curve got flatten, meaning the model found a solution.

KERAS PREPROCESSING

The Keras preprocessing layers API allows developers to build Keras-native input processing pipelines. These input processing pipelines can be used as independent preprocessing code in non-Keras workflows, combined directly with Keras models, and exported as part of a Keras SavedModel.

With Keras preprocessing layers, you can build and export models that are truly end-to-end: models that accept raw images or raw structured data as input; models that handle feature normalization or feature value indexing on their own.

- Core preprocessing layers
- Structured data preprocessing layers
- Image preprocessing layers
- Image data augmentation layers

- **EDIT DISTANCE**

Fast implementation of the edit distance(Levenshtein distance).
This library simply implements Levenshtein distance with C++ and Cython.

Above libraries only support strings. But Sometimes other type of objects such as list of strings(words). I support any iterable, only requires hashable object of it:

- **OPENCV**

OpenCV supports a wide variety of programming languages such as C++, Python, Java etc. Support for multiple platforms including Windows, Linux, and MacOS.

OpenCV Python is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays.

This makes it easier to integrate it with other libraries which use NumPy. For example, libraries such as SciPy and Matplotlib.

Next up on this OpenCV Python Tutorial blog, let us look at some of the basic operations that we can perform with OpenCV.

❖ **FEATURES OF THE LIBRARY**

- ➢ Read and write images
- ➢ Capture and save videos
- ➢ Process images (filter, transform)
- ➢ Perform feature detection
- ➢ Detect specific objects such as faces, eyes, cars, in the videos or images.
- ➢ Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

1. METHODOLOGY

Hand written text recognition:

We collected data from different age groups which had been written in paper so we need to divide them in to each letter which should be a single photo of each character so we had developed the code to divide them into images

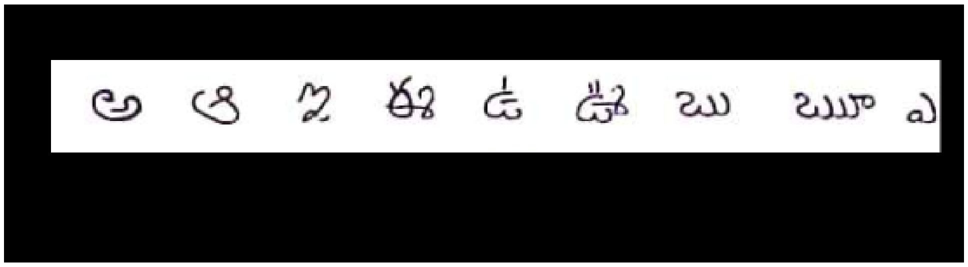


Figure 1: Input Image For word segmentation

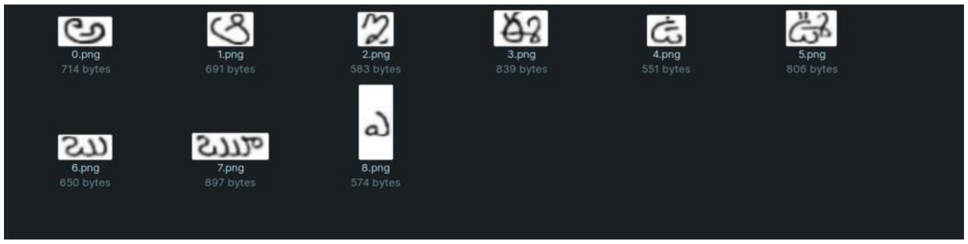


Figure 2: Output Images After word segmentation

After the word segmentation we need to get the boundaries of an image where the image is located in the picture so to get that we had developed the code to get the boundaries in that we will be having tool to draw the boundaries after that the boundaries will be saved into a ".txt "file

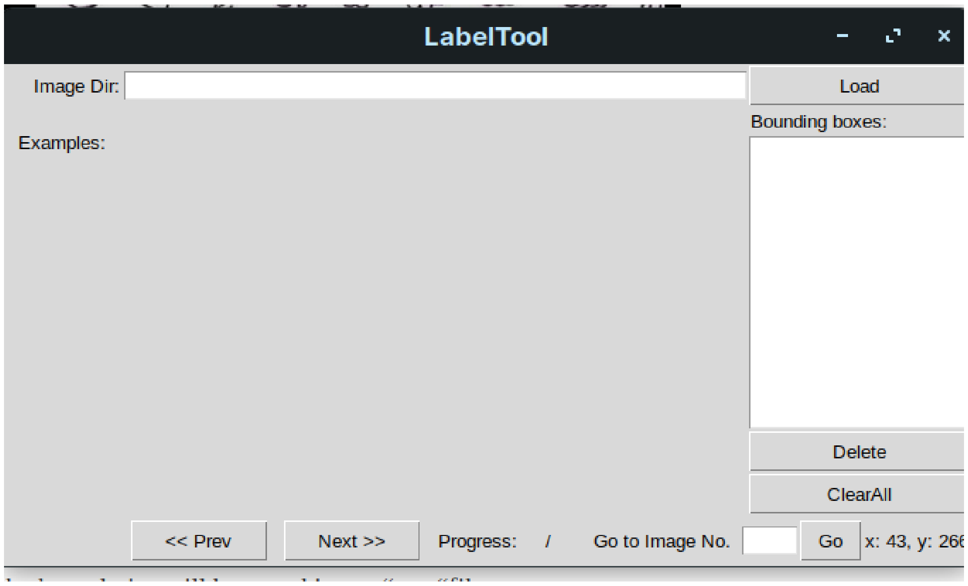


Figure 3: Bounding Box Interface

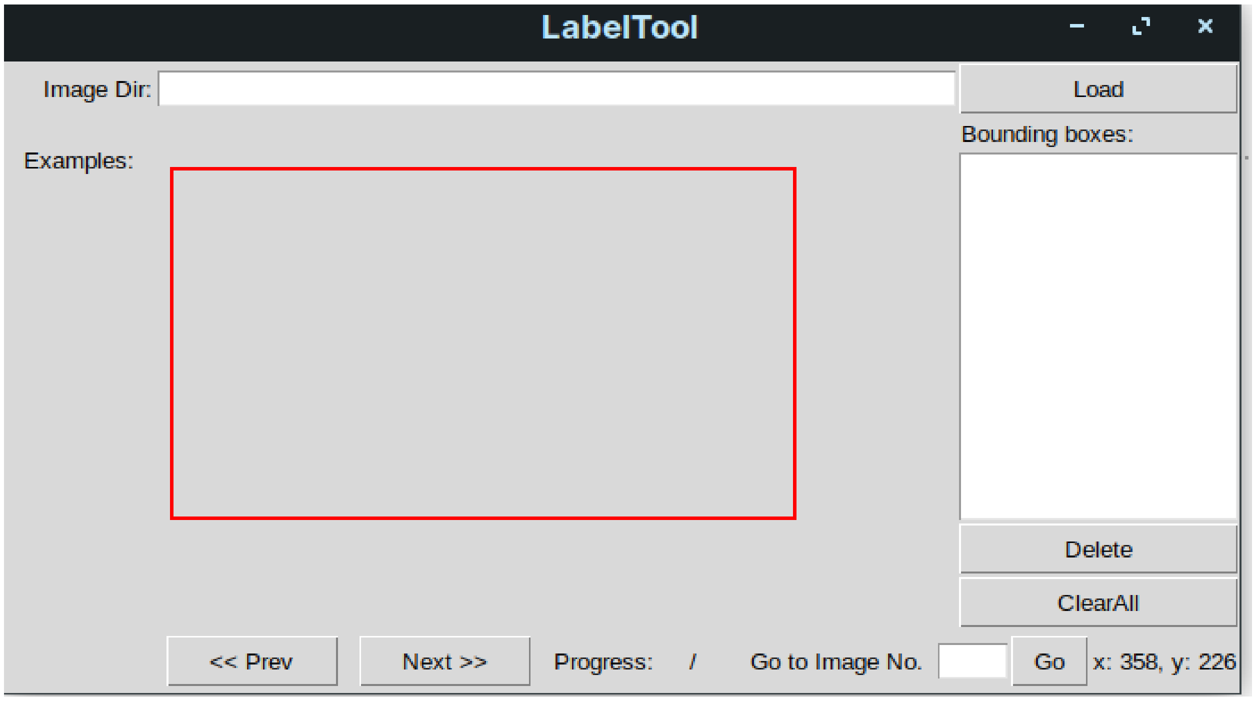
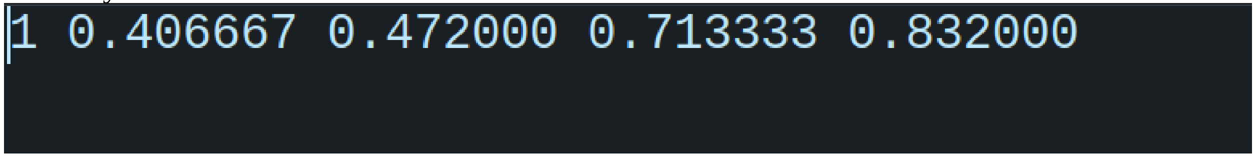


Figure 3.1: Bounding Box Interface with Area selection

If we load the directory will be opened and then we can select the boundaries of the character present in the image after capturing the boundaries we can have the stored boundaries in the label directory with the txt file

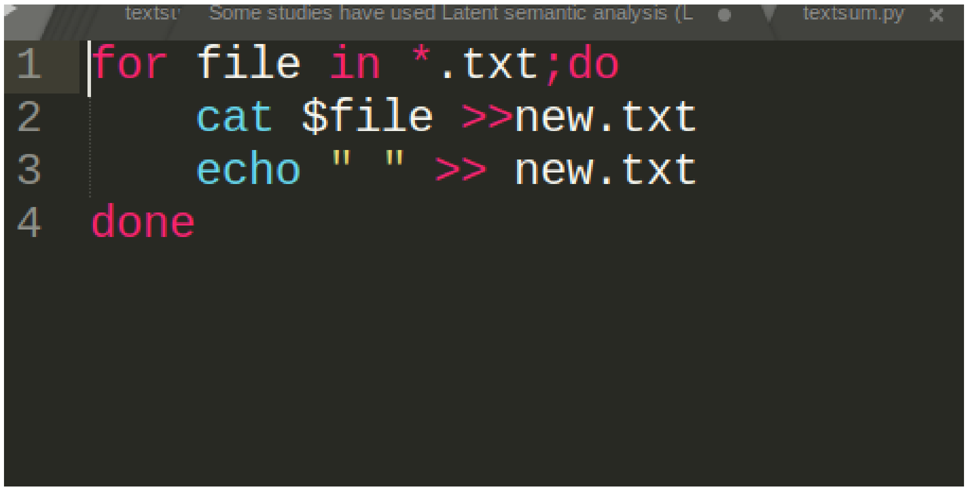


```
1 0.406667 0.472000 0.713333 0.832000
```

Figure 4: The output of the bounding box

we can have the x1,y1,x2,y2 as the boundries of the character present in the image

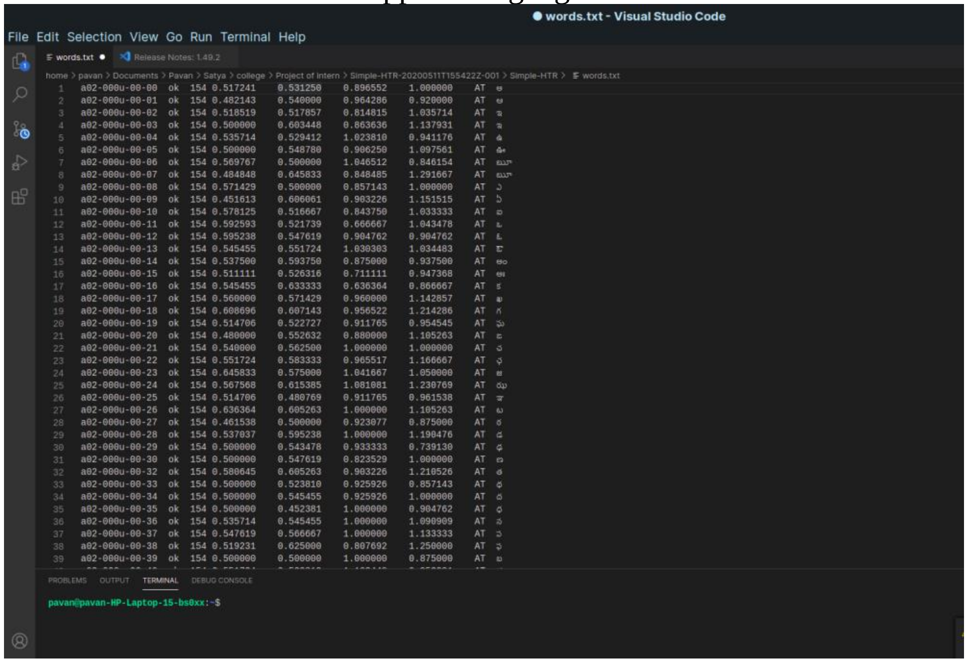
After that we will get all the txt files in the directory then we can combine all the files by using the shell scripting which will be very easy to finish it up



```
for file in *.txt;do
    cat $file >>new.txt
    echo " " >> new.txt
done
```

Figure 4: The output of the bounding box

This will combine all the single files into a final file and then we will append the file name into the filename and we need to also append the status as ok and then the constant as ok and the coordinates next we need to append language code as AT and the character present in that image


words.txt - Visual Studio Code
File Edit Selection View Go Run Terminal Help
words.txt
Release Notes: 1.49.2
home > pavan > Documents > Pavan > Satya > college > Project of Intern > Simple-HTR-20200511T155422Z-001 > Simple-HTR > words.txt
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE

```
a02-000u-00-00  ok  154 0.517241    0.531250    0.896552    1.000000    AT  అ
a02-000u-00-01  ok  154 0.482143    0.540000    0.964286    0.920000    AT  ఆ
a02-000u-00-02  ok  154 0.518519    0.517857    0.814815    1.035714    AT  ఇ
a02-000u-00-03  ok  154 0.500000    0.603448    0.863636    1.137931    AT  ఈ
a02-000u-00-04  ok  154 0.535714    0.529412    1.023810    0.941176    AT  ఉ
a02-000u-00-05  ok  154 0.500000    0.548780    0.906250    1.097561    AT  ఊ
a02-000u-00-06  ok  154 0.569767    0.500000    1.046512    0.846154    AT  ఋ
a02-000u-00-07  ok  154 0.484848    0.645833    0.848485    1.291667    AT  ౠ
a02-000u-00-08  ok  154 0.571429    0.500000    0.857143    1.000000    AT  ఎ
a02-000u-00-09  ok  154 0.451613    0.606061    0.903226    1.151515    AT  ఏ
a02-000u-00-10  ok  154 0.578125    0.516667    0.843750    1.033333    AT  ఐ
a02-000u-00-11  ok  154 0.592593    0.521739    0.666667    1.043478    AT  ఒ
a02-000u-00-12  ok  154 0.595238    0.547619    0.904762    0.904762    AT  ఓ
a02-000u-00-13  ok  154 0.545455    0.551724    1.030303    1.034483    AT  ఔ
a02-000u-00-14  ok  154 0.537500    0.593750    0.875000    0.937500    AT  అం
a02-000u-00-15  ok  154 0.511111    0.526316    0.711111    0.947368    AT  అః
a02-000u-00-16  ok  154 0.545455    0.633333    0.636364    0.866667    AT  క
a02-000u-00-17  ok  154 0.560000    0.571429    0.960000    1.142857    AT  ఖ
a02-000u-00-18  ok  154 0.608696    0.607143    0.956522    1.214286    AT  గ
a02-000u-00-19  ok  154 0.514706    0.522727    0.911765    0.954545    AT  ఘ
a02-000u-00-20  ok  154 0.480000    0.552632    0.880000    1.105263    AT  ఙ
a02-000u-00-21  ok  154 0.540000    0.562500    1.000000    1.000000    AT  చ
a02-000u-00-22  ok  154 0.551724    0.583333    0.965517    1.166667    AT  ఛ
a02-000u-00-23  ok  154 0.645833    0.575000    1.041667    1.050000    AT  జ
a02-000u-00-24  ok  154 0.567568    0.615385    1.081081    1.230769    AT  ఝ
a02-000u-00-25  ok  154 0.514706    0.480769    0.911765    0.961538    AT  ఞ
a02-000u-00-26  ok  154 0.636364    0.605263    1.000000    1.105263    AT  ట
a02-000u-00-27  ok  154 0.461538    0.500000    0.923077    0.875000    AT  ఠ
a02-000u-00-28  ok  154 0.537037    0.595238    1.000000    1.190476    AT  డ
a02-000u-00-29  ok  154 0.500000    0.543478    0.933333    0.739130    AT  ఢ
a02-000u-00-30  ok  154 0.500000    0.547619    0.823529    1.000000    AT  ణ
a02-000u-00-32  ok  154 0.580645    0.605263    0.903226    1.210526    AT  త
a02-000u-00-33  ok  154 0.500000    0.523810    0.925926    0.857143    AT  థ
a02-000u-00-34  ok  154 0.500000    0.545455    0.925926    1.000000    AT  ద
a02-000u-00-35  ok  154 0.500000    0.452381    1.000000    0.904762    AT  ధ
a02-000u-00-36  ok  154 0.535714    0.545455    1.000000    1.090909    AT  న
a02-000u-00-37  ok  154 0.547619    0.566667    1.000000    1.133333    AT  ప
a02-000u-00-38  ok  154 0.519231    0.625000    0.807692    1.250000    AT  ఫ
a02-000u-00-39  ok  154 0.500000    0.500000    1.000000    0.875000    AT  బ
```

```
pavan@pavan-HP-Laptop-15-bs0xx:~$
```

Figure 5:Representation of the Input to neural networks

To align the data into these patterns we had used many tools like excel to the file name and then we used the status as constant with the help of the editor shortcuts able to align them and then we used the coordinate into the excel and added them up after that we are using these to the feeding into the neural network

And after aligning the data we need to change the all the image file name into the specified format as mentioned this format is to have more directories and we can classify the data before training the data so that it will be more helpful to training the model so before training we are classifying it

Offline Handwritten Text Recognition (HTR) systems transcribe text contained in scanned images into digital text, an example is shown in Fig. 1. We will build a Neural Network (NN) which is trained on word-images prepared by us. 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.

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.

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).

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.

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. “చ”), or with duplicate characters (e.g. “ఛ”), or with character-properties such as loops (as contained in handwritten “జ”s or “ఛ”s).

.

❖ **PROCEDURE :**

I) SamplePreprocessor.py: prepares the images from the IAM dataset for the NN
II) DataLoader.py: reads samples, puts them into batches and provides an iterator-interface to go through the data
III) Model.py: creates the model as described above, loads and saves models, manages the TF sessions and provides an interface for training and inference
IV) main.py: puts all previously mentioned modules together
V) We only look at Model.py, as the other source files are concerned with basic file IO (DataLoader.py) and image processing (SamplePreprocessor.py).
VI) For each CNN layer, create a kernel of size k×k to be used in the convolution operation.
VII) Create and stack two RNN layers with 256 units each.
VIII) 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.
IX) 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.
X) 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.
XI) 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.

XII) Data augmentation: increase dataset-size by applying further (random) transformations to the input images

XIII) Remove cursive writing style in the input images (see DeslantImg)

XIV) Increase input size (if input of NN is large enough, complete text-lines can be used)

XV) Add more CNN layers

XVI) Replace LSTM by 2D-LSTM

XVII) Decoder: use token passing or word beam search decoding (see CTCWordBeamSearch) to constrain the output to dictionary words

XVIII) Text correction: if the recognized word is not contained in a dictionary, search for the most similar one

2. OBSERVATION AND DISCUSSION

For the first week I had been given with some basic tasks to understand the working the TensorFlow library. Those tasks included basic pre-processing functions. These tasks acted as a bridge to the further tasks I performed. I had used TensorFlow to work on the data, to modify and manipulate the data. Using the write functions, I printed the data in several other types.

I have used the tensor board library to plot the graphs from the processed python data. From keras pre-processing we had pe processed the data to feed into the model so that model will have easy representation

I have learnt new things such as how to build neural network and what is CNN and what is RNN and how the data plays an important role in training the model and I also I have learned how TensorFlow and keras can be used in the data augmentation and how the data can be replicated what are the pre necessary steps that are to be performed on the data. How to calculate the loss functions and what is optimizer how can it help increasing the accuracy and decrease the loss so that model can have more accuracy how optimizer plays a role in increasing the accuracy

The current booming technologies which I've learnt will be handy for me in the future. Now I can be able to classify, cluster, predict the datasets using the ML techniques using pandas library. The basic knowledge in bootstrap will help me to design and build websites in efficient and beautiful way.

3. CONCLUSION

The drawbacks of the platform were solved. We provided the user with better user experience. I have added additional features which will be helpful for the end user to access the data easily and also navigate with the features. I have contributed in building the dashboard and writing small pieces of bootstrap codes for the Dashboard. I’ve studied about the new concepts of plotly library, Flask API’s, how a website runs, what is MVC Architecture.

In a nutshell, this internship has been an excellent and rewarding experience. I can conclude that there have been a lot I’ve learnt from my work at Concent Solutions. Not only did I gain practical skills but I also had the opportunity to meet many fantastic people who will be able to help me with opportunities in the future. The atmosphere at the office was always welcoming. Two main things that I’ve learned are the importance of time-management skills and continuous learning skills.

Needless to say, the technical aspects of the work I’ve done are not flawless and could be improved provided enough time. Overall, my internship at Concent Solutions Pvt. Ltd. has been a success. I was able to gain many technical skills, work in a fantastic environment, and make connections that will last a lifetime.

REFERENCES

1) https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5 --- Hand written recognition
2) https://keras.io/api/preprocessing/ --- Keras
3) https://www.youtube.com/watch?v=DFKHh7_zzJc --- Tensorflow Tutorial

4) https://docs.opencv.org/master/d9/df8/tutorial_root.html --- OpenCV
5) https://algorithmia.com/blog/introduction-to-optimizers --- DOM
6) https://www.toptal.com/machine-learning/tensorflow-machine-learning-tutorial --- Tensorflow
7) https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html --- OpenCV
8)

APPENDIX – I

SAMPLE SOURCE CODE

1. SAMPLE CODE FOR Tensorflow IN PYTHON:

```
from __future__ import division
from __future__ import print_function
```

```
import sys
import argparse
import cv2
import editdistance
from DataLoader import DataLoader, Batch
from Model import Model, DecoderType
from SamplePreprocessor import preprocess



class FilePaths:
    "filenames and paths to data"
    fnCharList = '../model/charList.txt'
    fnAccuracy = '../model/accuracy.txt'
    fnTrain = '../data/'
    fnInfer = '../data/'
    fnCorpus = '../data/corpus.txt'



def train(model, loader):
    "train NN"
    epoch = 0 # number of training epochs since start
    bestCharErrorRate = float('inf') # best valdiation character error rate
    noImprovementSince = 0 # number of epochs no improvement of character error rate occured
    earlyStopping = 5 # stop training after this number of epochs without improvement
    while True:
        epoch += 1
        print('Epoch:', epoch)

        # train
        print('Train NN')
        loader.trainSet()
        while loader.hasNext():
            iterInfo = loader.getIteratorInfo()
            batch = loader.getNext()
            loss = model.trainBatch(batch)
            print('Batch:', iterInfo[0],'/', iterInfo[1], 'Loss:', loss)

        # validate
        charErrorRate = validate(model, loader)

        # if best validation accuracy so far, save model parameters
        if charErrorRate < bestCharErrorRate:
            print('Character error rate improved, save model')
            bestCharErrorRate = charErrorRate
            noImprovementSince = 0
            model.save()
            open(FilePaths.fnAccuracy, 'w').write('Validation character error rate of saved model: %f%%' %
(charErrorRate*100.0))
        else:
            print('Character error rate not improved')
            noImprovementSince += 1

        # stop training if no more improvement in the last x epochs
        if noImprovementSince >= earlyStopping:
            print('No more improvement since %d epochs. Training stopped.' % earlyStopping)
            break



def validate(model, loader):
    "validate NN"
    print('Validate NN')
    loader.validationSet()
    numCharErr = 0
    numCharTotal = 0
    numWordOK = 0
    numWordTotal = 0
    f=open("result.txt","w")
    while loader.hasNext():
        iterInfo = loader.getIteratorInfo()
        print('Batch:', iterInfo[0],'/', iterInfo[1])
        batch = loader.getNext()
        (recognized, _) = model.inferBatch(batch)

        print('Ground truth -> Recognized')
        for i in range(len(recognized)):
            numWordOK += 1 if batch.gtTexts[i] == recognized[i] else 0
            numWordTotal += 1
            dist = editdistance.eval(recognized[i], batch.gtTexts[i])
            numCharErr += dist
            numCharTotal += len(batch.gtTexts[i])
            print('[OK]' if dist==0 else '[ERR:%d]' % dist,'"' + batch.gtTexts[i] + '"', '->', '"' + recognized[i] +
'"')
            f.write("Actual Letter" +batch.gtTexts[i] +"recognized letter" +recognized[i] + "\n")

    # print validation result
    charErrorRate = numCharErr / numCharTotal
    wordAccuracy = numWordOK / numWordTotal
    print('Character error rate: %f%%. Word accuracy: %f%%.' % (charErrorRate*100.0,
wordAccuracy*100.0))
    return charErrorRate


def infer(model, fnImg):
    "recognize text in image provided by file path"
    img = preprocess(cv2.imread(fnImg, cv2.IMREAD_GRAYSCALE), Model.imgSize)
    batch = Batch(None, [img])
    (recognized, probability) = model.inferBatch(batch, True)
    print('Recognized:', '"' + recognized[0] + '"')
    print('Probability:', probability[0])


def main():
    "main function"
    # optional command line args
    parser = argparse.ArgumentParser()
    parser.add_argument('--train', help='train the NN', action='store_true')
    parser.add_argument('--validate', help='validate the NN', action='store_true')
    parser.add_argument('--beamsearch', help='use beam search instead of best path decoding', action='store_true')
    parser.add_argument('--wordbeamsearch', help='use word beam search instead of best path decoding', action='store_true')
    parser.add_argument('--dump', help='dump output of NN to CSV file(s)', action='store_true')

    args = parser.parse_args()

    decoderType = DecoderType.BestPath
    if args.beamsearch:
        decoderType = DecoderType.BeamSearch
    elif args.wordbeamsearch:
        decoderType = DecoderType.WordBeamSearch

    # train or validate on IAM dataset
    if args.train or args.validate:
        # load training data, create TF model
        loader = DataLoader(FilePaths.fnTrain, Model.batchSize, Model.imgSize, Model.maxTextLen)

        # save characters of model for inference mode
        open(FilePaths.fnCharList, 'w',encoding='utf-8').write(str().join(loader.charList))

        # save words contained in dataset into file
        open(FilePaths.fnCorpus, 'w',encoding='utf-8').write(str(' ').join(loader.trainWords + loader.validationWords))

        # execute training or validation
        if args.train:
            model = Model(loader.charList, decoderType)
            train(model, loader)
        elif args.validate:
            model = Model(loader.charList, decoderType, mustRestore=True)
            validate(model, loader)
    # infer text on test image
    else:
        loader = DataLoader(FilePaths.fnTrain, Model.batchSize, Model.imgSize, Model.maxTextLen)
        model = Model(loader.charList, decoderType, mustRestore=True)
        validate(model, loader)
        """print(open(FilePaths.fnAccuracy).read())
        model = Model(open(FilePaths.fnCharList).read(), decoderType, mustRestore=True, dump=args.dump)
        infer(model, FilePaths.fnInfer)"""



if __name__ == '__main__':
    main()
```

2. SAMPLE CODE FOR Bounding Box IN PYTHON:

```
# https://github.com/puzzledqs/BBox-Label-Tool
from __future__ import division
from tkinter import *
import tkinter.messagebox
from PIL import Image, ImageTk
import os
import glob
import random

# colors for the bboxes
COLORS = ['red', 'blue', 'yellow', 'pink', 'cyan', 'green', 'black']
# image sizes for the examples
SIZE = 350, 350

class LabelTool():
    def __init__(self, master):
```

```
self.imgclass = 1



# set up the main frame
self.parent = master
self.parent.title("LabelTool")
self.frame = Frame(self.parent)
self.frame.pack(fill=BOTH, expand=1)
self.parent.resizable(width = FALSE, height = FALSE)

# initialize global state
self.imageDir = ''
self.imageList= []
self.egDir = ''
self.egList = []
self.outDir = ''
self.cur = 0
self.total = 0
self.category = 0
self.imagename = ''
self.labelfilename = ''
self.tkimg = None

# initialize mouse state
self.STATE = {}
self.STATE['click'] = 0
self.STATE['x'], self.STATE['y'] = 0, 0

# reference to bbox
self.bboxIdList = []
self.bboxId = None
self.bboxList = []
self.hl = None
self.vl = None

# ----------------- GUI stuff ---------------------
# dir entry & load
self.label = Label(self.frame, text = "Image Dir:")
self.label.grid(row = 0, column = 0, sticky = E)
self.entry = Entry(self.frame)
self.entry.grid(row = 0, column = 1, sticky = W+E)
self.ldBtn = Button(self.frame, text = "Load", command = self.loadDir)
self.ldBtn.grid(row = 0, column = 2, sticky = W+E)

# main panel for labeling
self.mainPanel = Canvas(self.frame, cursor='tcross')
self.mainPanel.bind("<Button-1>", self.mouseClick)
self.mainPanel.bind("<Motion>", self.mouseMove)
self.parent.bind("<Escape>", self.cancelBBox)  # press <Espace> to cancel current bbox
self.parent.bind("s", self.cancelBBox)
self.parent.bind("a", self.prevImage) # press 'a' to go backforward
self.parent.bind("d", self.nextImage) # press 'd' to go forward
self.mainPanel.grid(row = 1, column = 1, rowspan = 4, sticky = W+N)

# showing bbox info & delete bbox
self.lb1 = Label(self.frame, text = 'Bounding boxes:')
self.lb1.grid(row = 1, column = 2,  sticky = W+N)
self.listbox = Listbox(self.frame, width = 22, height = 12)
self.listbox.grid(row = 2, column = 2, sticky = N)
self.btnDel = Button(self.frame, text = 'Delete', command = self.delBBox)
self.btnDel.grid(row = 3, column = 2, sticky = W+E+N)
self.btnClear = Button(self.frame, text = 'ClearAll', command = self.clearBBox)
self.btnClear.grid(row = 4, column = 2, sticky = W+E+N)
```

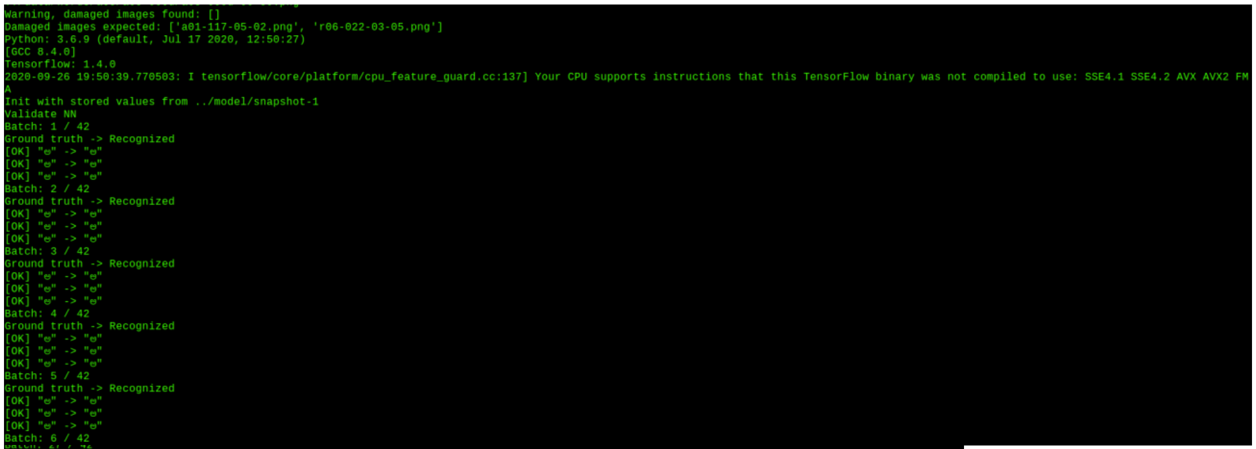
```
# control panel for image navigation
self.ctrPanel = Frame(self.frame)
self.ctrPanel.grid(row = 5, column = 1, columnspan = 2, sticky = W+E)
self.prevBtn = Button(self.ctrPanel, text='<< Prev', width = 10, command = self.prevImage)
self.prevBtn.pack(side = LEFT, padx = 5, pady = 3)
self.nextBtn = Button(self.ctrPanel, text='Next >>', width = 10, command = self.nextImage)
self.nextBtn.pack(side = LEFT, padx = 5, pady = 3)
self.progLabel = Label(self.ctrPanel, text = "Progress:     /    ")
self.progLabel.pack(side = LEFT, padx = 5)
self.tmpLabel = Label(self.ctrPanel, text = "Go to Image No.")
self.tmpLabel.pack(side = LEFT, padx = 5)
self.idxEntry = Entry(self.ctrPanel, width = 5)
self.idxEntry.pack(side = LEFT)
self.goBtn = Button(self.ctrPanel, text = 'Go', command = self.gotoImage)
self.goBtn.pack(side = LEFT)

# example pannel for illustration
self.egPanel = Frame(self.frame, border = 10)
self.egPanel.grid(row = 1, column = 0, rowspan = 5, sticky = N)
self.tmpLabel2 = Label(self.egPanel, text = "Examples:")
self.tmpLabel2.pack(side = TOP, pady = 5)
self.egLabels = []
for i in range(3):
    self.egLabels.append(Label(self.egPanel))
    self.egLabels[-1].pack(side = TOP)

# display mouse position
self.disp = Label(self.ctrPanel, text='')
self.disp.pack(side = RIGHT)

self.frame.columnconfigure(1, weight = 1)
self.frame.rowconfigure(4, weight = 1)
```

```
        # for debugging
##        self.setImage()
##        self.loadDir()

    def loadDir(self, dbg = False):

        self.imageDir = os.path.join('/home/pavan/Downloads/WordSegmentation-
20200430T050819Z-001/bounding_box-20200505T045842Z-001/bounding_box/images/',
str(self.imgclass))
        print(self.imageDir)
        self.imageList = glob.glob(os.path.join(self.imageDir, '*png'))
        self.imageList = sorted(self.imageList)
        if len(self.imageList) == 0:
            print('No .JPG images found in the specified dir!')
            return

        # default to the 1st image in the collection
        self.cur = 1
        self.total = len(self.imageList)

         # set up output dir
        self.outDir = os.path.join('/home/pavan/Downloads/WordSegmentation-
20200430T050819Z-001/bounding_box-20200505T045842Z-001/bounding_box/labels',
str(self.imgclass))
        if not os.path.exists(self.outDir):
            os.mkdir(self.outDir)



        filelist = glob.glob(os.path.join(self.imageDir, '*.png'))
        self.tmp = []
        self.egList = []
        #random.shuffle(filelist)
        for (i, f) in enumerate(filelist):
            if i == 3:
                break
            im = Image.open(f)
            r = min(SIZE[0] / im.size[0], SIZE[1] / im.size[1])
            new_size = int(r * im.size[0]), int(r * im.size[1])
            self.tmp.append(im.resize(new_size, Image.ANTIALIAS))
            self.egList.append(ImageTk.PhotoImage(self.tmp[-1]))
            self.egLabels[i].config(image = self.egList[-1], width = SIZE[0], height = SIZE[1])

        self.loadImage()
        print ('%d images loaded' %(self.total))

    def loadImage(self):
        # load image
        imagepath = self.imageList[self.cur - 1]
        self.img = Image.open(imagepath)
        self.tkimg = ImageTk.PhotoImage(self.img)
        self.mainPanel.config(width = max(self.tkimg.width(), 400), height =
max(self.tkimg.height(), 400))
        self.mainPanel.create_image(0, 0, image = self.tkimg, anchor=NW)
        self.progLabel.config(text = "%04d/%04d" %(self.cur, self.total))
        imgw = self.img.size[0]
        imgh = self.img.size[1]

        # load labels
        self.clearBBox()
        self.imagename = os.path.split(imagepath)[-1].split('.')[0]
        labelname = self.imagename + '.txt'
        self.labelfilename = os.path.join(self.outDir, labelname)
        bbox_cnt = 0
        if os.path.exists(self.labelfilename):
            with open(self.labelfilename) as f:
                line = [float(w) for w in f.read().split()]
                x1 = (line[1] - line[3]/2) * imgw
                y1 = (line[2] - line[4]/2) * imgh
                x2 = (line[1] + line[3]/2) * imgw
                y2 = (line[2] + line[4]/2) * imgh
                tmp = (x1,y1,x2,y2)
                self.bboxList.append(tuple(tmp))
                tmpId = self.mainPanel.create_rectangle(tmp[0], tmp[1], \
                                                        tmp[2], tmp[3], \
                                                        width = 2, \
                                                        outline = COLORS[(len(self.bboxList)-1) % len(COLORS)])
                self.bboxIdList.append(tmpId)
                self.listbox.insert(END, '(%d, %d) -> (%d, %d)' %(tmp[0], tmp[1], tmp[2], tmp[3]))
                self.listbox.itemconfig(len(self.bboxIdList) - 1, fg = COLORS[(len(self.bboxIdList) -
1) % len(COLORS)])

    def saveImage(self):
        box = self.bboxList[0]
        imgw = self.img.size[0]
        imgh = self.img.size[1]
        x = (box[0] + box[2]) / 2 / imgw
        y = (box[1] + box[3]) / 2 / imgh
        w = (box[2] - box[0]) / imgw
        h = (box[3] - box[1]) / imgh
        with open(self.labelfilename, 'w') as f:
            f.write("%d %f %f %f %f" % (self.imgclass, x, y, w, h))
        #print ('Image No. %d saved' %(self.cur))

    def mouseClick(self, event):
        if self.STATE['click'] == 0:
            self.STATE['x'], self.STATE['y'] = event.x, event.y
        else:
            x1, x2 = min(self.STATE['x'], event.x), max(self.STATE['x'], event.x)
            y1, y2 = min(self.STATE['y'], event.y), max(self.STATE['y'], event.y)
            self.bboxList.append((x1, y1, x2, y2))
            self.bboxIdList.append(self.bboxId)
            self.bboxId = None
            self.listbox.insert(END, '(%d, %d) -> (%d, %d)' %(x1, y1, x2, y2))
            self.listbox.itemconfig(len(self.bboxIdList) - 1, fg = COLORS[(len(self.bboxIdList) - 1)
% len(COLORS)])
        self.STATE['click'] = 1 - self.STATE['click']

    def mouseMove(self, event):
        self.disp.config(text = 'x: %d, y: %d' %(event.x, event.y))
        if self.tkimg:
            if self.hl:
                self.mainPanel.delete(self.hl)
            self.hl = self.mainPanel.create_line(0, event.y, self.tkimg.width(), event.y, width = 2)
            if self.vl:
                self.mainPanel.delete(self.vl)
            self.vl = self.mainPanel.create_line(event.x, 0, event.x, self.tkimg.height(), width = 2)
        if 1 == self.STATE['click']:
            if self.bboxId:
                self.mainPanel.delete(self.bboxId)
            self.bboxId = self.mainPanel.create_rectangle(self.STATE['x'], self.STATE['y'], \
                                                            event.x, event.y, \
                                                            width = 2, \
                                                            outline = COLORS[len(self.bboxList) % len(COLORS)])

    def cancelBBox(self, event):
        if 1 == self.STATE['click']:
            if self.bboxId:
```

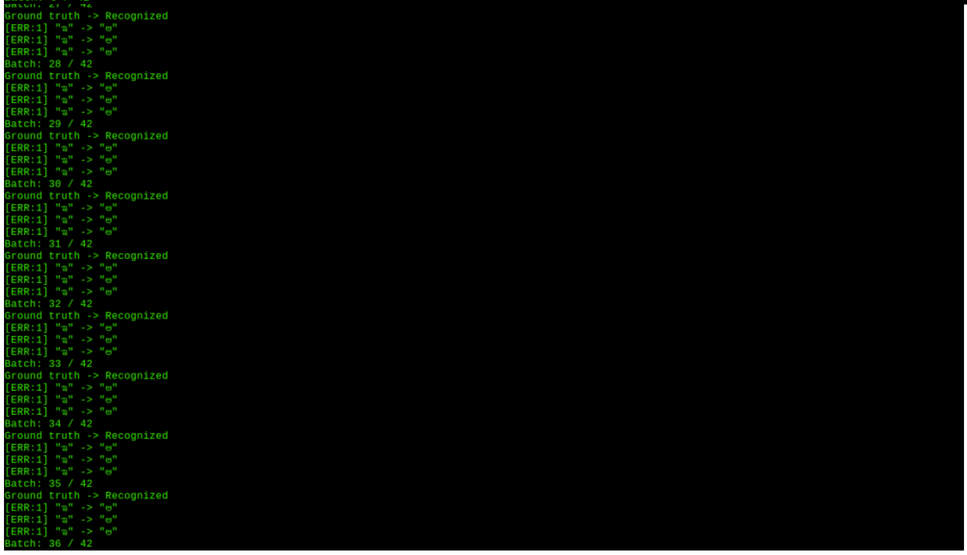
```
            self.mainPanel.delete(self.bboxId)
            self.bboxId = None
            self.STATE['click'] = 0

    def delBBox(self):
        sel = self.listbox.curselection()
        if len(sel) != 1 :
            return
        idx = int(sel[0])
        self.mainPanel.delete(self.bboxIdList[idx])
        self.bboxIdList.pop(idx)
        self.bboxList.pop(idx)
        self.listbox.delete(idx)

    def clearBBox(self):
        for idx in range(len(self.bboxIdList)):
            self.mainPanel.delete(self.bboxIdList[idx])
        self.listbox.delete(0, len(self.bboxList))
        self.bboxIdList = []
        self.bboxList = []

    def prevImage(self, event = None):
        self.saveImage()
        if self.cur > 1:
            self.cur -= 1
            self.loadImage()

    def nextImage(self, event = None):
        self.saveImage()
        if self.cur < self.total:
            self.cur += 1
            self.loadImage()
    def gotoImage(self):
        idx = int(self.idxEntry.get())
        if 1 <= idx and idx <= self.total:
            self.saveImage()
            self.cur = idx
            self.loadImage()

##    def setImage(self, imagepath = r'test2.png'):
##        self.img = Image.open(imagepath)
##        self.tkimg = ImageTk.PhotoImage(self.img)
##        self.mainPanel.config(width = self.tkimg.width())
##        self.mainPanel.config(height = self.tkimg.height())
##        self.mainPanel.create_image(0, 0, image = self.tkimg, anchor=NW)

if __name__ == '__main__':
    root = Tk()
    tool = LabelTool(root)
    root.resizable(width =  True, height = True)
    root.mainloop()
```

APPENDIX – II

Screenshots of the outputs



```
Warning, damaged images found: []
Damaged images expected: ['a01-117-05-02.png', 'r06-022-03-05.png']
Python: 3.6.9 (default, Jul 17 2020, 12:50:27)
[GCC 8.4.0]
Tensorflow: 1.4.0
2020-09-26 19:50:39.770503: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FM
A
Init with stored values from ../model/snapshot-1
Validate NN
Batch: 1 / 42
Ground truth -> Recognized
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
Batch: 2 / 42
Ground truth -> Recognized
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
Batch: 3 / 42
Ground truth -> Recognized
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
Batch: 4 / 42
Ground truth -> Recognized
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
Batch: 5 / 42
Ground truth -> Recognized
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
[OK] "ම" -> "ම"
Batch: 6 / 42
```



```
Ground truth -> Recognized
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
Batch: 28 / 42
Ground truth -> Recognized
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
Batch: 29 / 42
Ground truth -> Recognized
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
Batch: 30 / 42
Ground truth -> Recognized
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
Batch: 31 / 42
Ground truth -> Recognized
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
Batch: 32 / 42
Ground truth -> Recognized
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
Batch: 33 / 42
Ground truth -> Recognized
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
Batch: 34 / 42
Ground truth -> Recognized
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
Batch: 35 / 42
Ground truth -> Recognized
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
[ERR:1] "ඹ" -> "ම"
Batch: 36 / 42
```

```
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
Actual Letter☺recognized letter☺
```

Plain Text ▾ Tab Width: 8 ▾ Ln 37, Col 14 ▾ INS