VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI

Internship Report on

"AUTOMATE IDENTIFICATION AND RECOGNITION OF HANDWRITTEN TEXT FROM AN IMAGE"

Submitted in partial fulfillment for the award of Degree of,

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

By

LAVANYA D M
4AL18CS041

Under the Supervision of

Mrs. Deeksha M

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MOODBIDRI-574225, KARNATAKA

2021 – 2022

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI D.K. -574225

KARNATAKA

ALVA'S
Education Foundation®

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Internship report on "Automate Identification and Recognition of Handwritten Text from an Image" submitted by LAVANYA D M (4AL18CS041) is work done by her and is submitted during the academic year 2021 – 2022, in partial fulfilment of the requirements for the award of the degree of BACHELOR OF ENGINEERING in DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING of VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Internship report has been approved as it satisfies the academic requirements in respect of Internship work prescribed for the Bachelor of Engineering Degree.

Internship Guide	**Internship Coordinator**
Department of CS&E	**Department of CS&E**

Head of the Department
Department of CS&E

Examiners

Name of the Examiner	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

First I would like to thank **Mr. Debashis Roy,** ***TCS iON*** for giving me the opportunity to do an internship within the organization.

I would also like to thank all the people who worked along with me in **TCS iON** with their patience and openness they created an enjoyable working environment.

It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

I am highly indebted to Managing Trustee **Mr. Vivek Alva** and Principal **Dr.Peter Fernandes, Alva's** ***Institute of Engineering and Technology, Mijar*** for the facilities provided to accomplish this internship.

I thank my Head of the Department **Dr. Manjunath Kotari,** ***Professor, Department of Computer Science and Engineering*** for his constructive criticism throughout my internship.

I am thankful to Internship coordinator, **Mrs. Merlyn Melita Mathias**, ***Assistant Professor, Department of Computer Science and Engineering*** for her valuable guidance, support and advices to get and complete internship in above said organization.

I would like to thank our internship guide **Mrs. Deeksha M,** ***Assistant Professor, Department of Computer Science and Engineering*** for her constant help and support throughout the internship.

I am extremely grateful to my department staff members and friends who helped me in successful completion of this internship.

LAVANYA D M

4AL18CS041

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

1ST WEEK

DAY	DATE	NAME OF THE TOPIC/MODULE	COMPLETED
Saturday	17/10/2020	Introduction to Machine Learning	Yes
Sunday	18/10/2020	Need of AI Introduction to Deep Learning	Yes

2ND WEEK

DAY	DATE	NAME OF THE TOPIC/MODULE	COMPLETED
Monday	19/10/2020	Brief regarding industry project	Yes
Tuesday	20/10/2020	Working examples related to AI	Yes
Wednesday	21/10/2020	Learnt about the steps in the data science workflow within AI	Yes
Thursday	22/10/2020	Learnt about the common data science misconceptions	Yes
Friday	23/10/2020	Learnt about the supervised learning in ML	Yes
Saturday	24/10/2020	Continued Learning	Yes
Sunday	25/10/2020	Continued Learning	Yes

3RD WEEK

DAY	DATE	NAME OF THE TOPIC/MODULE	COMPLETED
Monday	26/10/2020	Learnt about comparison between training and inference	Yes
Tuesday	27/10/2020	Training versus testing data	Yes
Wednesday	28/10/2020	Modeling and validation	Yes
Thursday	29/10/2020	Learnt about accuracy and mean squared error	Yes
Friday	30/10/2020	Learnt about PyData Ecosystem	Yes
Saturday	31/10/2020	K Nearest Neighbor Algorithms	Yes
Sunday	01/11/2020	Continued Learning	Yes

4TH WEEK

DAY	DATE	NAME OF THE TOPIC/MODULE	COMPLETED
Monday	02/11/2020	Data types in python	Yes
Tuesday	03/11/2020	Under-fitting vs over-fitting curves in ML	Yes
Wednesday	04/11/2020	Algorithms in ML, software libraries and framework	Yes
Thursday	05/11/2020	Continued Learning	Yes
Friday	06/11/2020	Continued Learning	Yes
Saturday	07/11/2020	Deep learning frameworks	Yes
Sunday	08/11/2020	Continued Learning	Yes

5TH WEEK

DAY	DATE	NAME OF THE TOPIC/MODULE	COMPLETED
Monday	09/11/2020	Neural Networks Topics	Yes
Tuesday	10/11/2020	Algorithms of Neural Networks	Yes
Wednesday	11/11/2020	Data analytics acceleration libraries	Yes
Thursday	12/11/2020	NumPy and SciPy	Yes
Friday	13/11/2020	Continued Learning	Yes
Saturday	14/11/2020	Performance of processor	Yes
Sunday	15/11/2020	Continued Learning	Yes

6TH WEEK

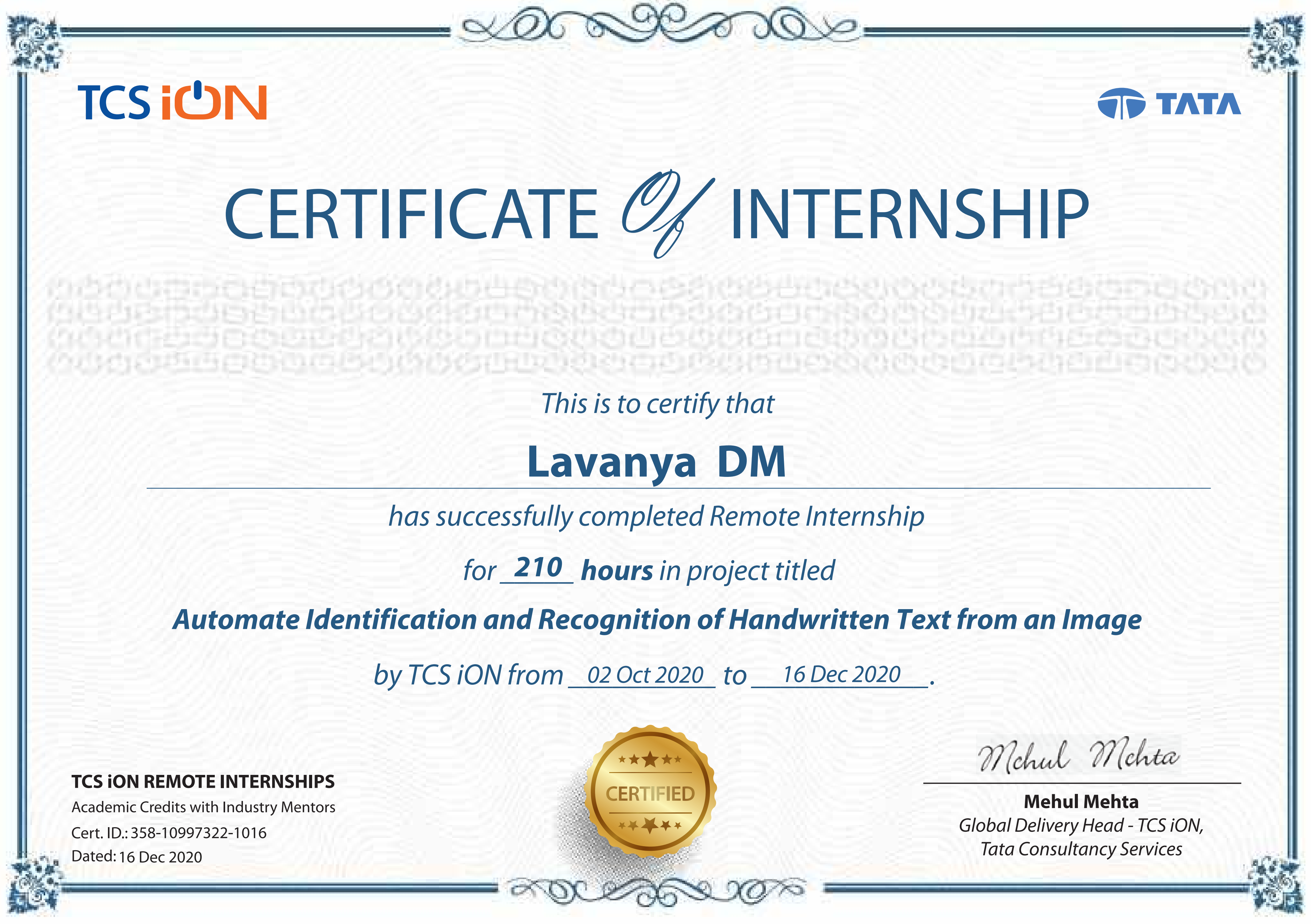
DAY	DATE	NAME OF THE TOPIC/MODULE	COMPLETED
Monday	16/11/2020	Brief working on ML	Yes
Tuesday	17/11/2020	Python Libraries used in ML	Yes
Wednesday	18/11/2020	Continued Learning	Yes
Thursday	19/11/2020	Learnt about OCR	Yes
Friday	20/11/2020	Continued Learning	Yes
Saturday	21/11/2020	Learnt about CNN and RNN	Yes
Sunday	22/11/2020	Continued Learning	Yes

7TH WEEK

DAY	DATE	NAME OF THE TOPIC/MODULE	COMPLETED
Monday	23/11/2020	Implementing the code	Yes
Tuesday	24/11/2020	Collecting the dataset required to train the model and the testing dataset	Yes
Wednesday	25/11/2020	Implementation of algorithm in code	Yes
Thursday	26/11/2020	Executing the code	Yes
Friday	27/11/2020	Executing the code	Yes
Saturday	28/11/2020	Executing the code	Yes
Sunday	29/11/2020	Testing the dataset for accurate result	Yes

8TH WEEK

DAY	DATE	NAME OF THE TOPIC/MODULE	COMPLETED
Monday	30/11/2020	Successfully done	Yes



TCS iON

TATA

CERTIFICATE *Of* INTERNSHIP

This is to certify that

Lavanya DM

has successfully completed Remote Internship

for **210** ***hours*** *in project titled*

Automate Identification and Recognition of Handwritten Text from an Image

by TCS iON from *02 Oct 2020* *to* *16 Dec 2020*.

TCS iON REMOTE INTERNSHIPS
Academic Credits with Industry Mentors
Cert. ID.: 358-10997322-1016
Dated: 16 Dec 2020

CERTIFIED

Mehul Mehta

Mehul Mehta
Global Delivery Head - TCS iON,
Tata Consultancy Services

COMPANY PROFILE

TCS iON is a strategic unit of Tata Consultancy Services focused on Manufacturing Industries (SMB), Educational Institutions and Examination Boards. TCS iON provides technology by means of a unique IT-as-a-Service model, offering end-to-end business solutions. It caters to the needs of multiple industry segments, through innovative, easy-to-use, secured, integrated, hosted solutions in a build-as-you-grow, pay-as-you-use business model. TCS iON serves its clients with the help of best practices gained through TCS' global experience, domestic market reach, skills, and delivery capabilities. TCS iON's Cloud Based Solution is highly modular, scalable and configurable giving businesses and educational institutions the benefits of increased efficiencies, faster go to market, predictability of technology as well as spend and better business results.

ABSTRACT

Identification and recognition of handwritten text from an image. It is based on enhancement of optical character recognition system. Optical character recognition or optical character reader is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image. An optical character recognition problem is basically a type of image-based sequence recognition problem. And for sequence recognition problem, most suited neural networks are recurrent neural networks (RNN) while for an image-based problem most suited are convolution neural networks (CNN). To cop up with the OCR problems we need to combine both of these CNN and RNN. The purpose of OCR in text identification and recognition from an image is to extract handwritten data to digital data. So, one can easily handle this digital data by editing, adding new information in that text. OCR is developing field of Computer Vision, Pattern Recognition and Artificial Intelligence.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

TCS iON is a strategic unit of Tata Consultancy Services focused on Manufacturing Industries (SMB), Educational Institutions and Examination Boards. TCS iON provides technology by means of a unique IT-as-a-Service model, offering end-to- end business solutions. It caters to the needs of multiple industry segments, through innovative, easy-to-use, secured, integrated, hosted solutions in a build-as-you-grow, pay- as-you-use business model. TCS iON serves its clients with the help of best practices gained through TCS' global experience, domestic market reach, skills, and delivery capabilities. TCS iON's Cloud Based Solution is highly modular, scalable and configurable giving businesses and educational institutions the benefits of increased efficiencies, faster go to market, predictability of technology as well as spend and better business results.

Optical Character Recognition (OCR) market size is expected to be USD 13.38 billion by 2025 with a year on year growth of 13.7 %. This growth is driven by rapid digitization of business processes using OCR to reduce their labor costs and to save precious man hours. Although OCR has been considered a solved problem there is one key component of it, Handwriting Recognition (Handwriting OCR) or Handwritten Text Recognition (HTR) which is still considered a challenging problem statement. The high variance in handwriting styles across people and poor quality of the handwritten text compared to printed text pose significant hurdles in converting it to machine readable text. Nevertheless it's a crucial problem to solve for multiple industries like healthcare, insurance and banking.

Recent advancements in Deep Learning such as the advent of transformer architectures have fast-tracked our progress in cracking handwritten text recognition. Recognizing handwritten text is termed Intelligent Character Recognition (ICR) due to the fact that the algorithms needed to solve ICR need much more intelligence than solving generic OCR.

In this article we will be learning about the task of handwritten text recognition, it's intricacies and how we can solve it using deep learning techniques.

Offline Text Recognition includes the science of recognizing both the human written font as well as the system-generated font. With the technology advancing, there arose an ardent need for combining conventional records into a digitized one, thereby, eliminating redundancies and diminishing the communication chain. With the world gravitating towards absolute digitization, there is a high demand for a Handwritten Text Recognition system. However, the challenge of implementing this system lies in the fact that handwritten words vary in characteristics such as slant and rounded letters, diacritic dots, crossbars, and humped letters.

A good handwriting recognition system must accurately identify the distorted characters and hence find the most plausible words.

The implementation of this project is done in three steps namely, line segmentation, word segmentation, and text recognition. Line segmentation is an important part of handwritten text recognition where the initial stage involves converting the images into greyscale, then inverted binary images are obtained followed by dilation of those images, and finally, boundary boxes are drawn over each line. Word segmentation is done by the scale-space technique. This process segments text lines obtained as the output of line segmentation into separate words which is used by the recognition system to recognize the text.

The objective of our proposed research work is to analyse the complexities involved in recognizing handwritten text by using both CNN and RNN and to calculate the loss using the Connectionist temporal classification (CTC) network. The other objectives are to test the performance evaluation of the CNN-RNN method compared with the conventional ones and to arrive at a method that can save time, enable faster processing, and reduce the probability of errors. An improvement in the man-to machine interactions in many applications occurs by utilizing the advanced automation processes involved in the handwritten text recognition systems.

CHAPTER 2

SYSTEM ANALYSIS

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

2.1 Requirement Specification

To be used efficiently, all computer software needs certain hardware components other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computersystems than technological advancements.

2.1.1 Hardware Requirements

Hardware components deal with the basic hardware requirement to develop andto run any system. It includes processor, memory etc.

Hardware Constraints:

- PROCESSOR: 1.4GHz Intel core i5
- RAM: 16 GB
- SSD/HDD: 1 TB
- GPU

2.1.2 Software Requirements

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Software Constraints:

- OPERATING SYSTEM: Windows, Linux
- LANGUAGE: Python, Tensorflow 2.0, Keras 2.3.0, OpenCV, Numpy, Scikit library
- TOOLS: VS Code, Anaconda, Jupyter Notebook

2.2 Libraries Used For Implementation

To implement project we have used Kaggle dataset (IAM dataset), Google Colab Virtual GPU as the platform to run the model, Tensorflow 2.0, Keras 2.3.0, OpenCV, Numpy, Scikit libraries and Jupyter Notebook.

2.3 Platforms Used For Implementation

- VS Code
- Anaconda
- Jupyter Notebook

CHAPTER 3

OBJECTIVE ACHIEVEMENT

The objective of the project is to train a model to achieve the best possible performance learning from our annotated dataset. The ultimate goal is a model reaching human-level performance. Such a model can be applied to visual recognition tasks.

Model iteration trains models with the annotated dataset and then evaluates the model's performance. If a model performs great, we can deploy it into production. If the performance is poor, we analyze the root causes to find ways to improve it.

Best Practices for Model Training:

Training neural network models is a time consuming process with many decisions:

- What image transformations should I use for data augmentation?
- Which model architecture to use?
- What metrics do I need to evaluate my model?
- Should I use ImageNet pretrained weights on my oil leakage images?
- What hyperparameters do I tune?

Before model training, review the dataset to build a qualitative understanding of the images. After reviewing 100 or more images, major patterns and variance will gradually surface.

- Where are the target defects in the image often located?
- If images are to be cropped to be smaller, which region should be retained?
- What is the pixel size of the smallest defects?
- Is global context needed to determine defects or local features within a 64 x64 pixel window?
- Are the defect class labels easily distinguishable from each other and free of ambiguities?
- Is a class label used less than half of the time?

3.1 Skills Learnt

- **Character Recognition Algorithms**

The algorithms used in character recognition can be divided into three categories: Image Pre-processing, Feature Extraction, and Classification. They are normally used in sequence – image pre-processing helps makes feature extraction a smoother process, while feature extraction is necessary for correct classification. Here's how they work:

- **Image pre-processing**

Image pre-processing is crucial in the recognition pipeline for correct character prediction. These methods typically include noise removal, image segmentation, cropping, scaling, and more. The recognition system first accepts a scanned image as an input. The images can be in JPG or BMT format. Digital capture and conversion of an image often introduces noise, which makes it hard to identify what is actually a part of the object of interest. Considering the problem of character recognition, we want to reduce as much noise as possible, while preserving the strokes of the characters, since they are important for correct classification.

- **Segmentation**

In the segmentation stage, a sequence of characters is segmented into a sub-image of an individual character. Each character is resized into 30×20 pixels.

- **Classification and Recognition**

This stage is the decision making stage of the recognition system. The classifier contains two hidden layers, using a log sigmoid activation function to train the algorithm.

- **Feature extraction**

The features of input data are the measurable properties of observations, which is used to analyse or classify these instances of data. The task of feature extraction is to identify relevant features that discriminate the instances that are independent of each other.

- **Neural Network System for Continuous Handwritten Word Recognition**

A method for continuous handwritten word recognition is derived when the word is segmented into triplets (containing 3 letters). Two subsequent triplets have 2 common letters. The biggest challenge for recognition systems is to perform operations on a continuous word. In this, each word is subdivided into triplets, each containing three letters. Figure 10a shows triplet “aba” and figure 10b shows triplet “ban”. Two neighbour triplets always contain two common letters which represent the overlapping between letters. This kind of overlapping results is a higher recognition rate.

- **Optical Character Recognition**

OCR stands for Optical Character Recognition. It is used to recognize text inside images, such as scanned documents and photos. OCR technology is used to convert virtually any kind of image containing written text into machine-readable text data.

OCR Technology became popular in the early 1990s while attempting to digitise historic newspapers. Since then the technology has undergone several improvements. Nowadays solutions deliver almost perfect OCR accuracy. Advanced methods like Zonal OCR are used to automate complex document based workflows.

The advantage of OCR software handwriting recognition is considerable. Now, with advances in technology, it is possible to scan a page of structured handwritten text and the converting engine can quickly use OCR software handwriting recognition to convert it to a machine-readable document.

- **Convolutional Neural Network(CNN) :**

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision. CNN can run directly on a underdone image and do not need any preprocessing. A convolutional neural network is a feed forward neural network, seldom with up to 20. The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer. CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces. The agenda for this sphere is to activate machines to view the world as humans do, perceive it in a alike fashion and even use the knowledge for a multitude of duty such as image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

- **Recurrent Neural Network(RNN)**

RNN are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

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.

3.3 CHALLENGES

- Detection of cursive letters
- The variation in Strokes
- The clarity of images
- The alignment of text in an image
- Efficiency of model

CHAPTER 4

IMPLEMENTATION

We can break the implementation of CRNN network into following steps:

1. Setting up kaggle
2. Collecting Dataset
3. Preprocessing Data
4. Creating Network Architecture
5. Defining Loss Function
6. Training Model
7. Testing and Prediction

1) Setting Up kaggle:

This is optional method to run this model. This method is only for use of GPU on Google Colab fastly. If one wants to use GPU on local machine then this step is not required. If we upload the datset on Google Drive and use this data for training purpose it takes 462 seconds per epoch and if we upload same dataset on kaggle and used on Google Colab it takes nearly 224 seconds per epoch. It means it takes half the time as compared to Google Drive so I used kaggle to load dataset in Google Colab.

2) Collecting Dataset:

This is one of the main task to implement our model effectively. The features of data provided in the project guidelines matches with IAM dataset. IAM dataset have cursive handwriting, poor image quality generated from scanned documents and skewed images. So, I decided to go with IAM dataset for this project. This is large dataset total of 1.09 GB (115320-Images). Here I have used only 7850 images for the training set and 876 images for validation dataset. This data contains text image segments which look like images shown below:

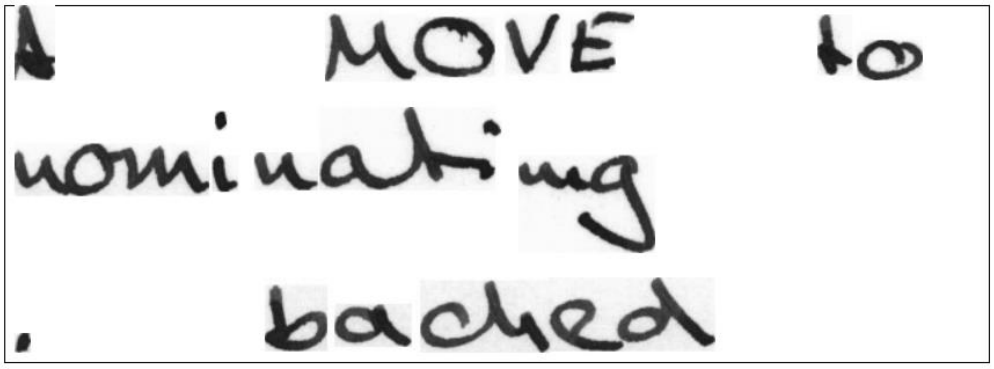


Fig 4.1 Dataset

3) Preprocessing Data:

Now we have our dataset, to make it acceptable for our model we have to use preprocessing of our dataset. We have to preprocess both input images and output labels. To Preprocess input images we have to follow the below steps:

- Read the image and convert it into a gray-scale image.
- Make each image of size (128, 32) using padding.
- Expand image dimension as (128,32,1) to make it compatible with the input shape of architecture
- Normalize the image pixel values by dividing it with 255.

To preprocess the output labels follow the below steps:

- Read the text from the words.txt file. This file contains every image text.
- Encode each character of a word into some numerical value by creating a function.
- Compute the maximum length from words and pad every output label to make it of the same size as the maximum length. This is done to make it compatible with the output shape of our RNN architecture.

In preprocessing we need further two lists. One is for label length and other is for input length to our RNN. This two lists are important for our CTC loss. Label length is the length of each output text label and input length is the same for each input to the LSTM layer which is 31 in our architecture.

4) Creating Network Architecture:

Input shape for our architecture having an input image of height 32 and width 128. Here we used seven convolution layers of which 6 are having kernel size (3, 3) and the last one is of size (2.2). And the number of filters is increased from 64 to 512 layer by layer. Two max-pooling layers are added with size (2, 2) and then two max-pooling layers of size (2, 1) are added to extract features with a larger width to predict long texts. Also, we used batch normalization layers after fifth and sixth convolution layers which accelerates the training process. Then we used a lambda function to squeeze the output from conv layer and make it compatible with LSTM layer. Then used two Bidirectional LSTM layers each of which has 128 units.

5) Defining Loss Function:

Now we have prepared model architecture, the next thing is to choose a loss function. In this text recognition problem, we will use the CTC loss function. CTC loss is very helpful in text recognition problems. It helps us to prevent annotating each time step and help us to get rid of the problem where a single character can span multiple time step which needs further processing if we do not use CTC. A CTC loss function requires four arguments to compute the loss, predicted outputs, ground truth labels, input sequence length to LSTM and ground truth label length.

6) Training Model:

To train the model I used Adam optimizer. Also, we can use Keras callbacks functionality to save the weights of the best model on the basis of validation loss. In model.compile(), I have only taken y_pred and neglected y_true. This is because I have already taken labels as input to the model earlier. Labels as input to the model earlier. Now train our model on 7850 training images and 876 validation images.

7) Testing and Prediction:

Our model is now trained with 7850 images. Now its time to test the model. We cannot use our training model because it also requires labels as input and at test time we cannot have labels. So to test the model we will use ” act_model ” that we have created earlier which takes only one input: test images. As our model predicts the probability for each class at each time step, we need to use some transcription function to convert it into actual texts. Here I used the CTC decoder to get the output text. I used Jaro Distance & Ratio method to test accuracy.

This approach I used for solving this problem. I also used other approaches but when we increase the complexity of image then this models not good so I continued with the CRNN model.

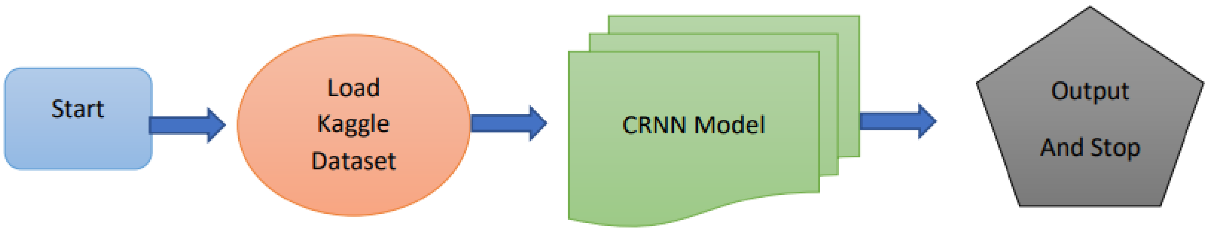


Fig 4.2 Block diagram

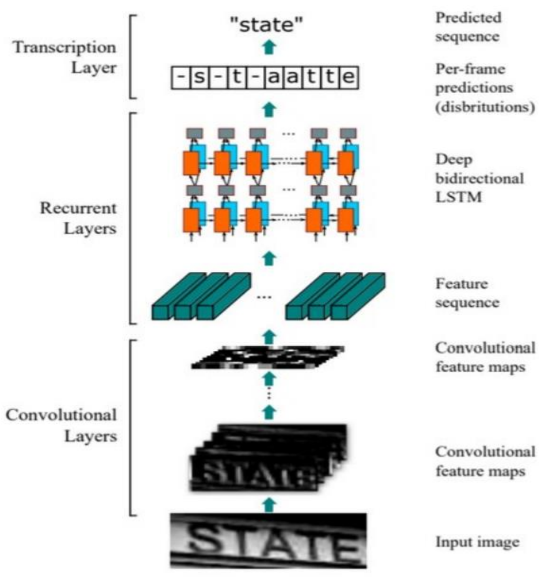


Fig 4.3 CRNN model

The architecture of this model consists of three parts:

1. Convolutional layers, which extract a feature sequence from the input image
2. Recurrent layers, which predict a label distribution for each frame
3. Transcription layer, which translates the per-frame predictions into the final label sequence.

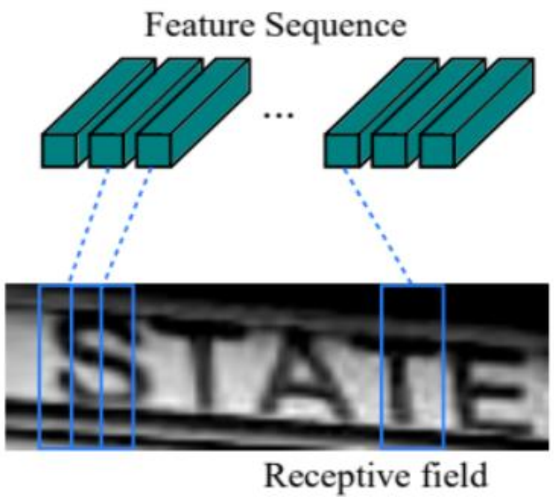


Fig 4.4 Feature sequence diagram

The receptive field. Each vector in the extracted feature sequence is associated with a receptive field on the input image, and can be considered as the feature vector of that field.

LSTM:

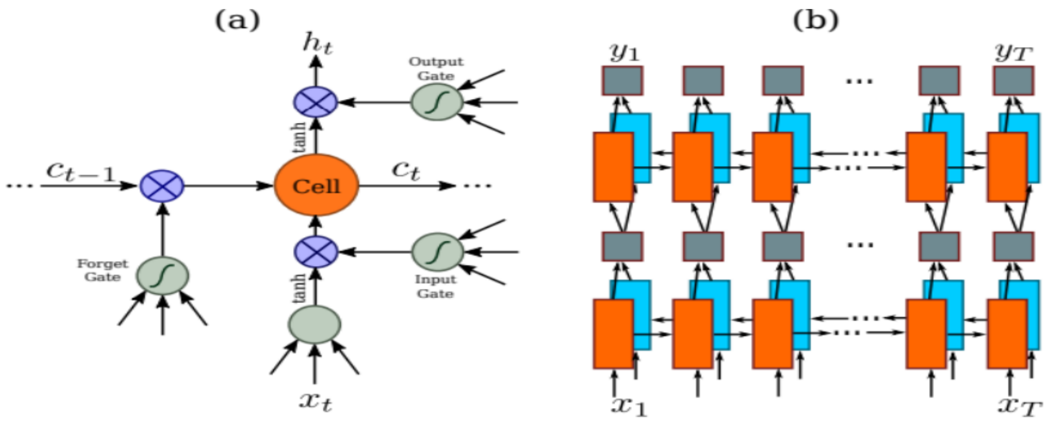


Fig 4.5 Bidirectional LSTM

(a) The structure of a basic LSTM unit. An LSTM consists of a cell module and three gates, namely the input gate, the output gate and the forget gate.

(b) The structure of deep bidirectional LSTM we use in our paper. Combining a forward (left to right) and a backward (right to left) LSTMs results in a bidirectional LSTM. Stacking multiple bidirectional LSTM results in a deep bidirectional LSTM.

Algorithm:

Model = CNN + RNN + CTC loss

Our model consists of three parts:

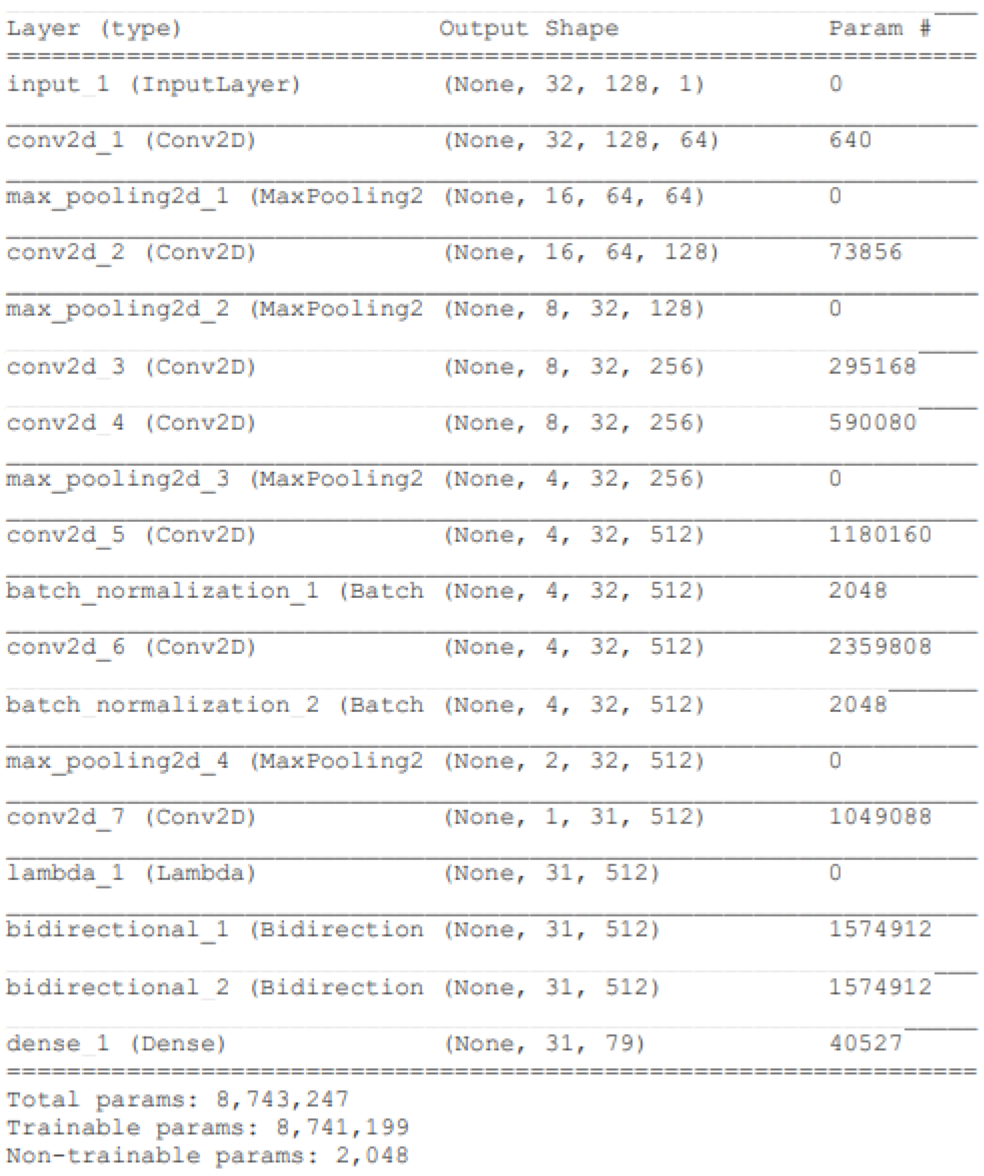
1. The convolutional neural network to extract features from the image

2. Recurrent neural network to predict sequential output per time-step

3. CTC loss function which is transcription layer used to predict output for each time step.

Model Architecture:

Here is the model architecture that we used.



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 32, 128, 1)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 128, 64)       640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 64, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 64, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 32, 128)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 32, 256)        295168
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 32, 256)        590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 32, 256)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 32, 512)        1180160
_________________________________________________________________
batch_normalization_1 (Batch (None, 4, 32, 512)        2048
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 4, 32, 512)        2359808
_________________________________________________________________
batch_normalization_2 (Batch (None, 4, 32, 512)        2048
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 32, 512)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 1, 31, 512)        1049088
_________________________________________________________________
lambda_1 (Lambda)            (None, 31, 512)           0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 31, 512)           1574912
_________________________________________________________________
bidirectional_2 (Bidirection (None, 31, 512)           1574912
_________________________________________________________________
dense_1 (Dense)              (None, 31, 79)            40527
=================================================================
Total params: 8,743,247
Trainable params: 8,741,199
Non-trainable params: 2,048
```

Fig 4.6 Model Architecture

Algorithm of CRNN Model:

1. Input shape for our architecture having an input image of height 32 and width 128.

2. Here we used seven convolution layers of which 6 are having kernel size (3, 3) and the last one is of size (2.2). And the number of filters is increased from 64 to 512 layer by layer.

3. Two max-pooling layers are added with size (2, 2) and then two max-pooling layers of size (2, 1) are added to extract features with a larger width to predict long texts.

4. Also, we used batch normalization layers after fifth and sixth convolution layers which accelerates the training process.

5. Then we used a lambda function to squeeze the output from conv layer and make it compatible with LSTM layer.

6. Then used two Bidirectional LSTM layers each of which has 128 units. This RNN layer gives the output of size (batch_size, 31, 63). Where 63 is the total number of output classes including blank character.

CHAPTER 5

RESULTS

The algorithm is able to detect and segment handwritten text from an image. The model successfully able to detect maximum words, which makes it about 91.72% accurate while implementation and testing.

The input image having the handwritten text is given as following:

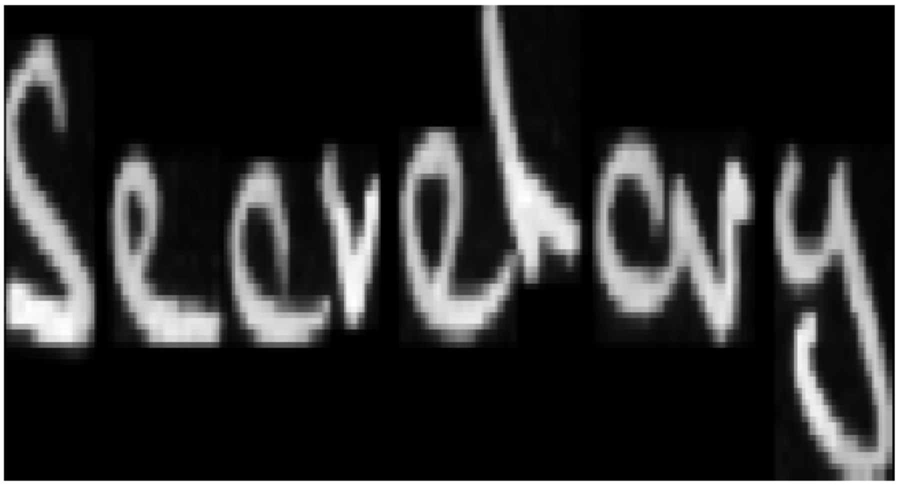
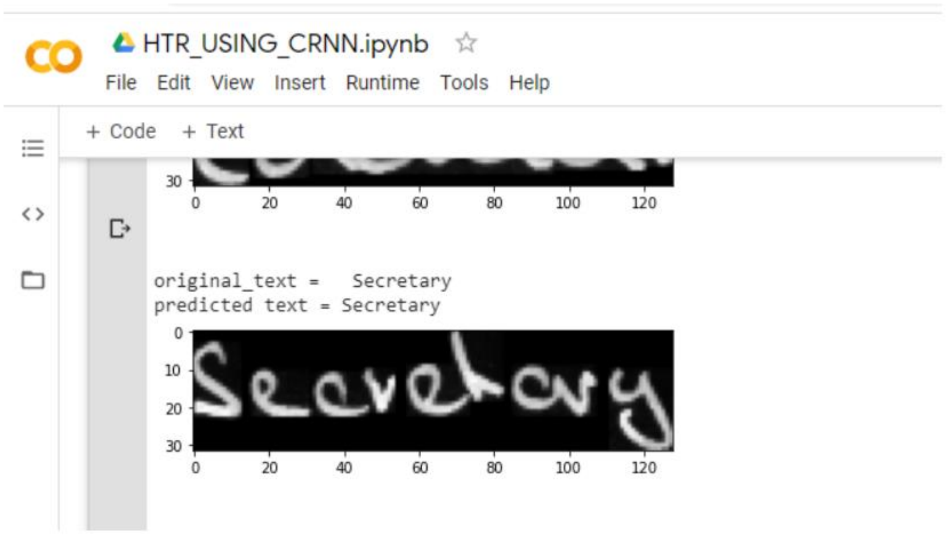


Fig 5.1 Input of handwritten text

The Model pre-process the image removes the noise and using CRNN algorithm predict the text.

Extracted Output: Secretary

As we can see the model is accurate and successfully able to extract the handwritten text. The model identify and recognize the text from the image as follows.



```
original_text =   Secretary
predicted text = Secretary
```

Fig 5.2 Extracted output of the given input

CHAPTER 6

CONCLUSION

Machine learning has been increasingly popular and it has begun to be applied to various areas ranging from facial recognition to target detection, OCR included. We continuously applied target detection networks of different types such as Faster and RCNN to the OCR area, in order to detect texts. Compared with traditional text recognition, text recognition embedded by deep learning is more accurate with stable noise immunity and robustness. It is able to resist influences such as changes in backgrounds. The end-to-end network of machine learning is more accurate when used in the area of text detection, compared with traditional image processing and cutting, and character classification enabled by machine learning.

REFERENCES

[1] https://arxiv.org/pdf/1507.05717.pdf

[2] https://software.intel.com/content/www/us/en/develop/training/course-artificial-intelligence.html

[3] https://software.intel.com/content/www/us/en/develop/training/course-machine-learning.html

[4] https://www.python-course.eu/machine_learning.php

[5] https://numpy.org/doc/

[6] https://software.intel.com/en-us/ai/courses/deep-learning

[7] https://www.tensorflow.org/tutorials/images/classification

[8] https://www.tensorflow.org/tutorials/text/text_classification_rnn

[9] https://www.tensorflow.org/tutorials/images/cnn

[10] https://www.tensorflow.org/tutorials/keras/classification

[11] https://www.tensorflow.org/tutorials

[12] https://pandas.pydata.org/pandas-docs/version/0.15/tutorials.html

[13] http://www.fki.inf.unibe.ch/databases/iam-handwriting-database/download-the-iam-handwriting-database

[14] https://towardsdatascience.com/setting-up-kaggle-in-google-colab-ebb281b61463

[15] http://www.fki.inf.unibe.ch/DBs/iamDB/data/words/