DIGITALIZED PAPER CORRECTION USING OPENCV

A PROJECT REPORT

Submitted by

DEEPAVARSHNI.K 211418104044

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University,Chennai)

MAY 2022

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"DIGITALIZED PAPER ORRECTION USING OPENCV"** is the bonafide work of "**DEEPAVARSHNI.K[211418104044]"** who carried out the project work under my supervision.

SIGNATURE

Dr.S.MURUGAVALLI,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Dr.L.JABASHEELA
SUPERVISOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above mentioned students were examined in End Semester project viva-voce held on _____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

I DEEPAVARSHNI.K(211418104044) hereby declare that this project report titled “DIGITALIZED PAPER CORRECTION USING OPENCV” under the guidance of Dr.L.JABASHEELA is the original work done by me and I have not plagiarized or submitted to any other degree in any other university .

DEEPAVARSHNI.K

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the Computer science and engineering Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide** Dr.L.JABASHELLA and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

ABSTRACT

In spite of rapid evolution of digital technologies, an enormous number of applications still depend on the utilization of paper based medium.It is a awfully difficult task to manually process the handwritten documents thanks to kinds of handwritten scripts and lack of associated language dictionary to interpret documents. Most of the big companies still as small-scale industries want to automate the method of script recognition. the massive challenge is to create machines recognize the hand-written scripts. Humans can recognize handwritten or hand-printed words after gaining knowledge of a selected language. within the same way, machines should be trained to acknowledge the handwritten scripts. This process of transferring human knowledge to computers should be automated.In this project a paper correction format in digitalized way using opencv is proposed.The system needs the answer paper in image format. The user has to enter the solution with number of solutions and the answers written by the student should be highlighted inside a box.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER NO. **TITLE** **PAGE NO.**

CHAPTER 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Nowadays everything has become online oriented .In This system I have proposed a concept to detect the handwritten text and to verify whether the solution written by the coed is correct or not. Handwriting text recognition (HTR) is that the process of adjusting handwritten characters or phrases into a format that the PC understands. it's a vigorous network of educational researchers studying it for the past few years as advances during this subject help to automate differing kinds of habitual tactics and job work. An example may be a painstaking seek of a scientific document inside tons of handwritten ancient manuscripts by a historian, which is requires an enormous amount of your time. Converting these manuscripts right into a virtual layout using HTR algorithms could permit the historian to search out the information within some seconds. The key advances in HTR in mail

communication are primarily geared toward finding solutions to the issues of recognition of the region of interest within the images, text segmentation, elimination of interference when working with text background noises, like missing or ambiguous bits, spots on paper, detection of skew. In handwriting recognition the device interprets the user's handwritten characters or words into a format that the computer understands (e.g., Unicode text). The input device typically comprises a stylus and a touch-sensitive screen. There are many levels of HWR, starting from the recognition of simplified individual characters to the recognition of whole words and sentences of cursive handwriting

1.2 PROBLEM DEFINITION

This paper proposes a technique to digitalize the paper correction and makes the teacher’s work easy.This project acts as an integration so we are able to boost any existing system so no new UI is required to interact.This can be easily utilized by all the users with little knowledge on computer.The image file may be anywhere within the system to access the system advantage. We've got experienced many fundamental breakthroughs within the past years and yet, despite all this progress within the underlying technologies, HTR is much from being a solved problem. However, given recent trends on how briskly we are able to move forward, this might change relatively soon. And just to say that additionally. Already today we are able to provide companies meaningful technology which will help reduce manual efforts significantly. Using higher-quality images that are easier for character recognition as inputs.Removing background using machine learning algorithms or improved photography practices.Developing more advanced recognition algorithms to

manage handwriting OCR tasks more accurately.Designing documents in an OCR-friendly way.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

1. Shilpa Mangesh Pande, Bineet Kumar Jha,Character Recognition System for Devanagari Script Using Machine Learning Approach,2021[1]

The proposed research work attempts to automate the character recognition system for Nagari script using various machine learning classifiers like Decision Tree classifier, Nearest Centroid classifier, K Nearest Neighbors classifier, Extra Trees classifiers and Random Forest classifier

2. Naragudem Sarika, Nageswararao Sirisala, Muni Sekhar Velpuru ,CNN based Optical Character Recognition and Applications,2021[2]

To translate text in a picture into text format, the Optical Character Recognition system is employed. There are three key aspects of OCR approach: pre-processing, character recognition, character segmentation and presentation of

knowledge. Convolutional Neural Network could be a deep learning method which is employed for character recognition.

3. Ayan Kumar Bhunia, Shuvozit Ghose, Amandeep Kumar, Pinaki Nath Chowdhury, Aneeshan Sain, Yi-Zhe,Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021[3]

This work is on the assumption that there is always a new style that is drastically different, and that we will only have very limited data during testing to perform adaptation. This results in creates a commercially viable solution -- being exposed to the new style, the model has the best shot at adaptation, and the few-sample nature makes it practical to implement.

4. Daniyar Nurseitov, Kairat Bostanbekov, Maksat Kanatov, Anel Alimova, Abdelrahman Abdallah, Galymzhan Abdimanap, Classification of Handwritten Names of Cities and Handwritten Text Recognition using Various Deep Learning Models, Advances in Science, Technology and Engineering Systems. 5, 934-943 ,2021[4]

The first model uses deep convolutional neural networks (CNNs) for feature extraction and a fully connected multilayer perceptron neural network (MLP) for word classification. The second model, called SimpleHTR, uses CNN and recurrent neural network (RNN) layers to extract information from images.

5. Daniyar Nurseitov, Kairat Bostanbekov, Maksat Kanatov,Classification of handwritten names of cities and Handwritten text recognition using various deep learning models,2020[5]

The first model uses deep convolutional neural networks (CNNs) for feature extraction and a totally connected multilayer perceptron neural network (MLP) for word classification. The second model, called Simple HTR, uses CNN and recurrent neural network (RNN) layers to extract information from images

6. A Scalable Handwritten Text Recognition System, R. Reeve Ingle, Yasuhisa Fujii, Thomas Deselaers, Jonathan Baccash, Ashok C. Popat,2019[6]

This paper addresses three problems in building such systems: data, efficiency, and integration. Firstly, one in all the largest challenges is obtaining sufficient amounts of top quality training data. We address the matter by using online handwriting data collected for an oversized scale production online handwriting recognition system. We describe our image data generation pipeline and study how online data is wont to build HTR models

7. Tianwei Wang,Yuanzhi Zhu, Lianwen Jin, Canjie Luo, Xiaoxue Chen,Decoupled Attention Network for Text Recognition,2019[7]

The system proposes a decoupled attention network (DAN), which decouples the alignment operation from using historical decoding results. DAN is a good, flexible and robust end-to-end text recognizer which consists of three components

8. Rajib Ghosh, Chinmaya Panda,Handwritten Text Recognition in Bank Cheques,2018[8]

There are four stages within the system: cropping the image at a particular location; segmentation of handwritten lines, words and characters; feature extraction from individual characters and digits using Histogram of Oriented Gradients (HOG) method and gray Level Co-occurrence Matrix (GLCM) texture features; recognition of isolated characters and digits using the Support Vector Machine (SVM) based classification process that ensures correct recognition .

CHAPTER 3

SYSTEM ANALYSIS

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

- In the existing system they calculate the digit character recognition to identify the letters.
- This took too long time to produce the result as the performance was not good.
- It’s a time consuming process.

3.2 PROPOSED SYSTEM

- In the proposed system this not acts as an individual system but we can integrate in such the way we need,the application can be combined with some other application for fast action.
- Th staff first needs to enter the path where the image file is present.
- They also need to enter the answers with number of answers by checking the question paper for evaluating to correct or not.
- The system evaluates the answer to be correct or not and then produces the result.

3.3 FEASIBILITY STUDY

TECHNICAL FEASIBILITY

- Index.py:This uses the installation of many modules needed by the system and checks the answer to it to produce the result.
- Onlyrectangle.py:This function scans the answer written inside the box alone.
- Main.py:This provides the skeleton part of the evaluation done.

SOCIAL FEASIBILITY

- This system helps the evaluator work easy.
- The file can be stored anywhere inside the computer

- It saves a lot of time.

ECONOMIC FEASIBILITY

- This system is an open source architecture.
- It acts like an integration system where it can be added to any existing contents.

3.4 HARDWARE ENVIRONMENT

- OS-Windows 10
- RAM-4.00 GB
- Processor Intel(R) Core(TM) i3-2328M CPU @ 2.20GHz

3.5 SOFTWARE ENVIRONMENT

- PyCharm IDE

CHAPTER 4

SYSTEM DESIGN

CHAPTER 4

SYSTEM DESIGN

4.1 ER DIAGRAM

ER-Diagram may be a representation of information that describes how data is communicated and associated with one another. Any object, like entities, attributes of an entity, sets of relationship, and other attributes of relationship, may be characterized with the assistance of the ER diagram. Entities: they're represented using the rectangle-shaped box. These rectangles are named with the entity set they represent.

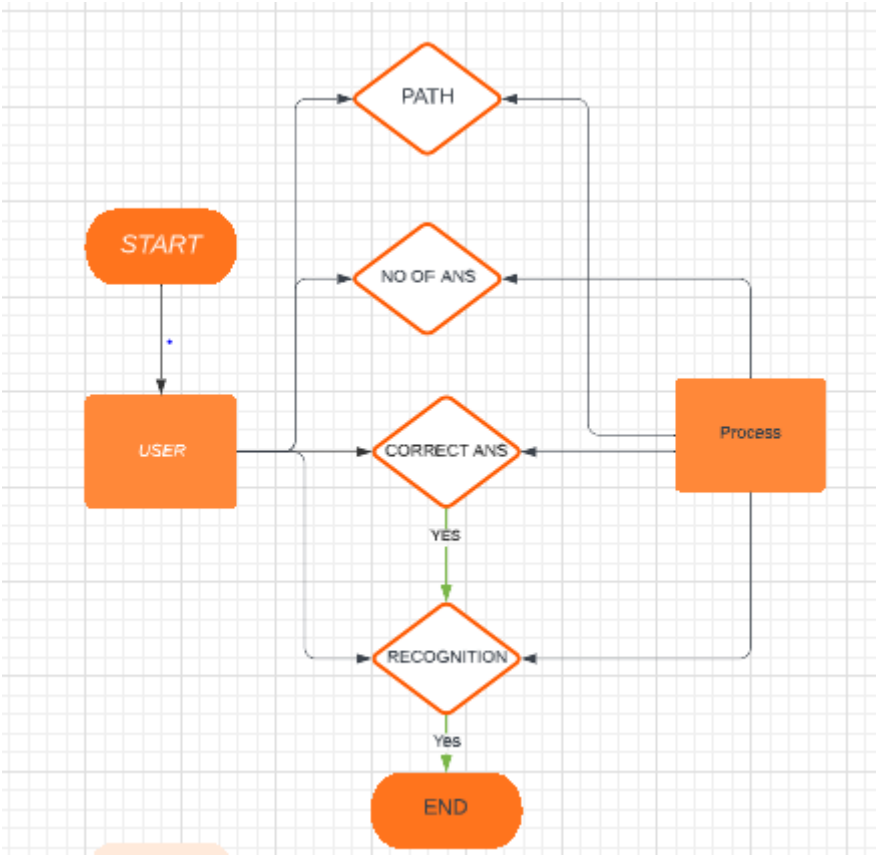

Figure 4.1 Entity Diagram for Digitalized paper correction

4.2 DATASET

IAM Dataset, which has variable length ground-truth targets. Each sample within the dataset is a picture of some handwritten text, and its corresponding target is that the string present within the image. The IAM Dataset is widely used across many OCR benchmarks, so we hope this instance can function an honest place to begin for building OCR systems. The IAM Handwriting Database contains sorts

of handwritten English text which might be wont to train and test handwritten text recognizers and to perform writer identification and verification experiments. The IAM Handwriting Database 3.0 is structured as follows:

- 657 writers contributed samples of their handwriting
- 1'539 pages of scanned text
- 5'685 isolated and labeled sentences
- 13'353 isolated and labeled text lines
- 115'320 isolated and labeled words

4.3 SYSTEM FLOW DIAGRAM

The system flow diagram is one of the graphical representations of the flow of data in a system in software engineering. The diagram consists of several steps that identify where the input is coming to the system and output going out of the system. With the help of the diagram, it is possible to control the event decisions of the system and how data is flowing to the system. Therefore, the system flow

diagram is basically a visual representation of data flow, excluding the minor parts and including the major parts of the system in a sequential manner.

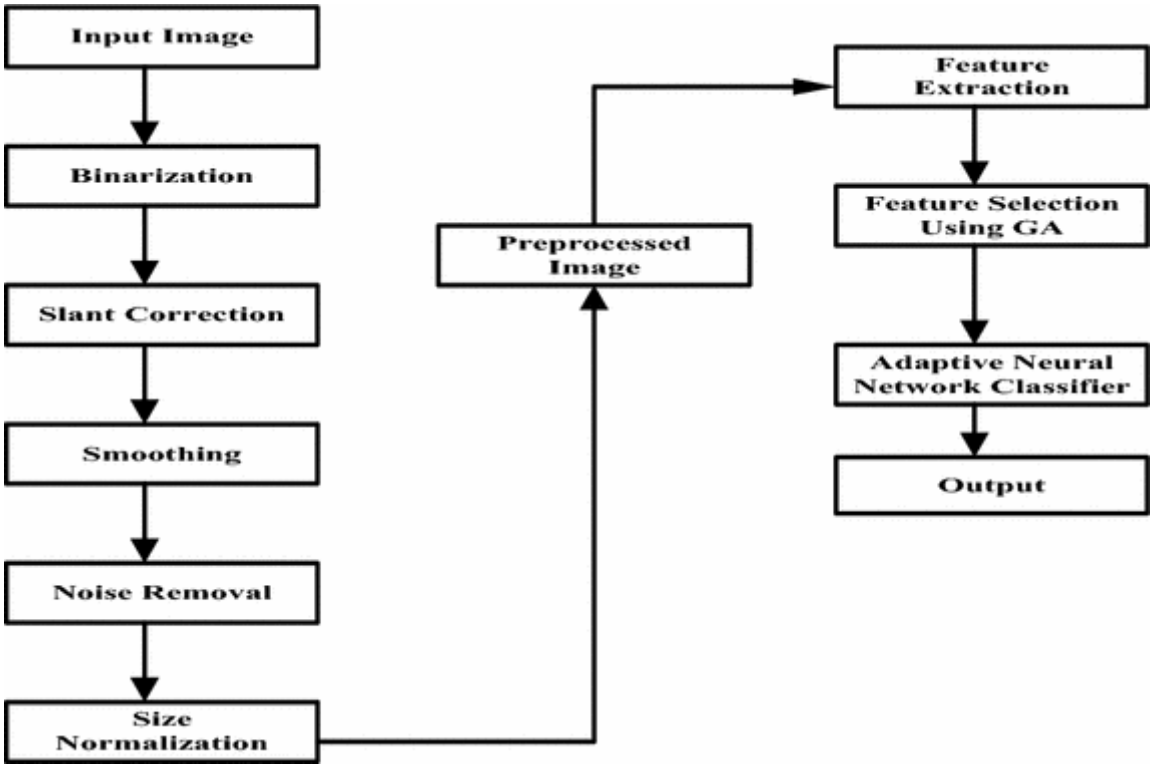

Figure 4.3 System Flow Diagram for Digitalized Paper Correction

4.3 UML DIAGRAMS

4.3.1 USE CASE DIAGRAM

Use case diagrams are typically developed within the early stage of development and other people often apply use case modeling for the subsequent purposes:

• Specify the context of a system

• Capture the necessities of a system

• Validate a systems architecture

• Drive implementation and generate test cases

• Developed by analysts along with domain experts

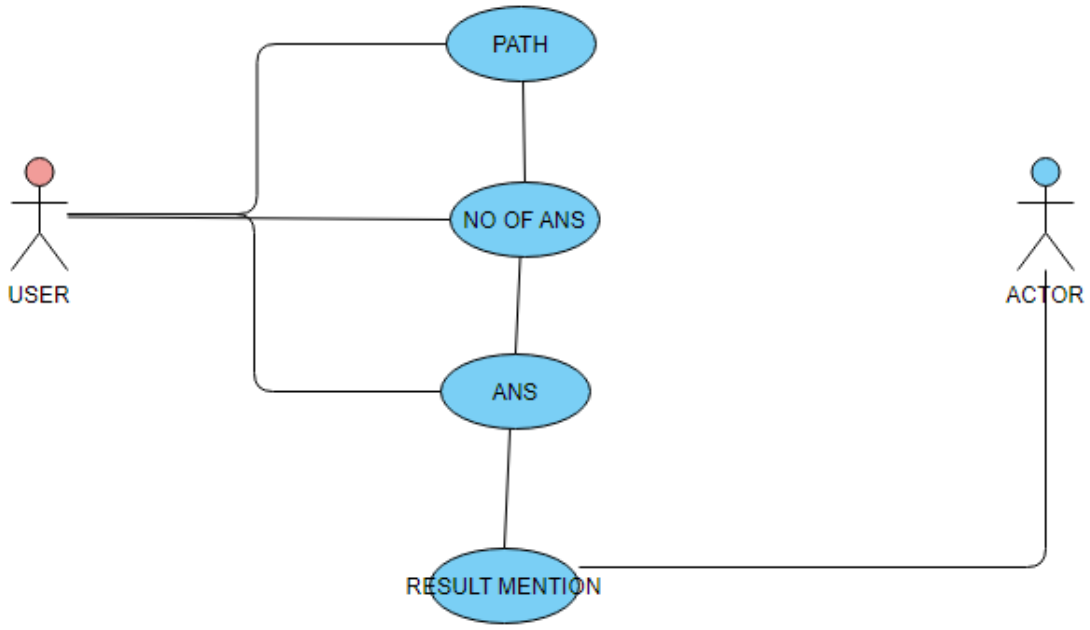

Figure 4.4 Use Case Diagram for Digitalized Paper Correction

4.3.2 CLASS DIAGRAM

The UML Class diagram is a graphical notation used to construct and visualize object oriented systems. A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's:

classes,

their attributes,

operations (or methods),

and the relationships among objects.

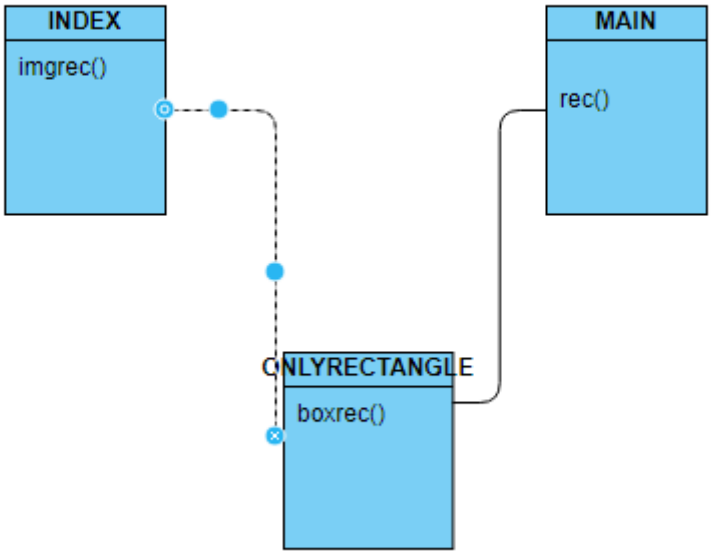

Figure 4.4 Class Diagram for Digitalized Paper Correction

4.3.3 ACTIVITY DIAGRAM

Activity diagram is another important behavioral diagram in UML diagram to explain dynamic aspects of the system. Activity diagram is basically a sophisticated version of flow chart that modeling the be due one activity to a different activity.

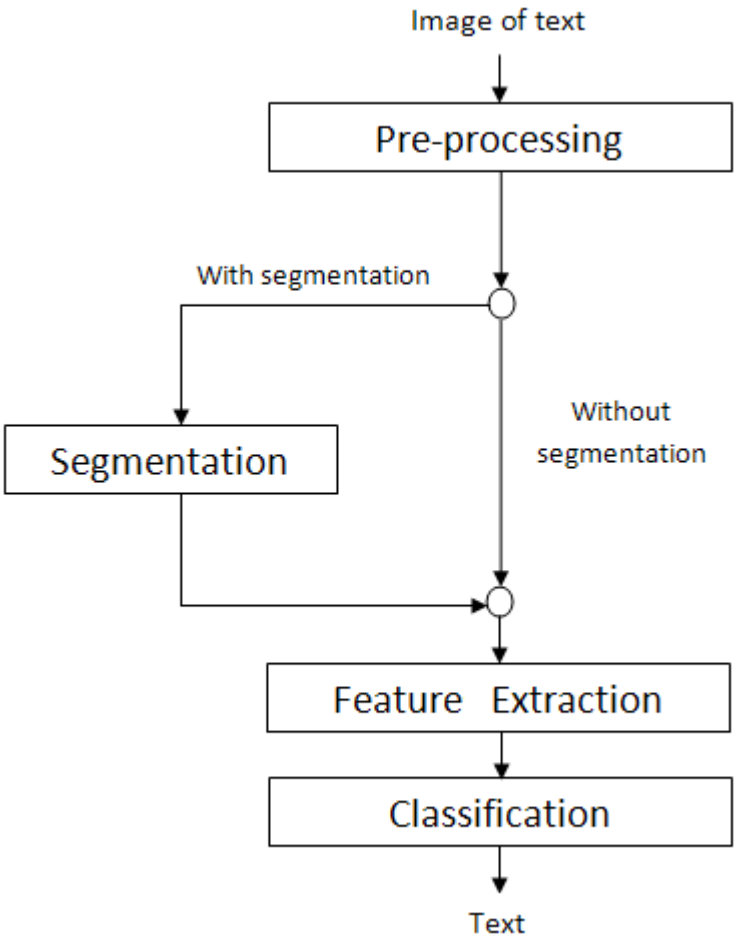

Figure 4.4 Activity Diagram for Digitalized paper correction

CHAPTER 5

SYSTEM ARCHITECTURE

CHAPTER 5

SYSTEM ARCHITECTURE

5.1 ARCHITECTURAL DIAGRAM

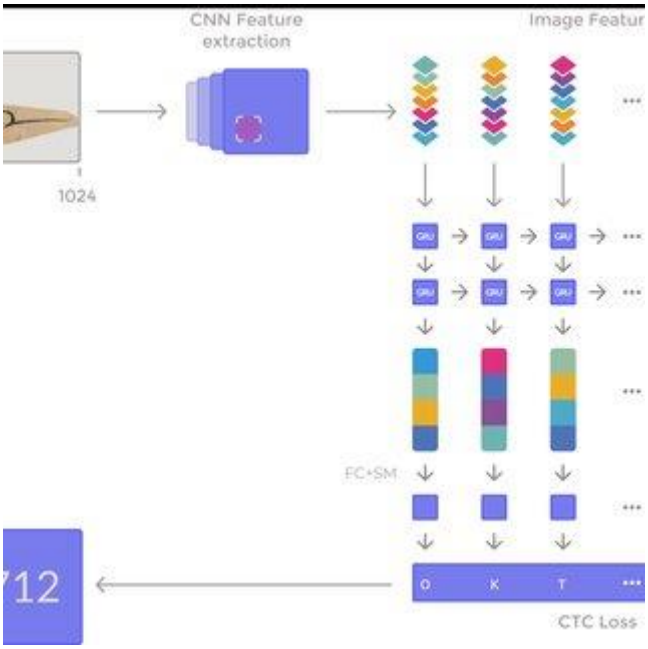

Figure 5.1 System Architecture Diagram

5.2 MODULES DESIGN SPECIFICATION

- Data Entry Module
- Recognition Module

MODULE DESCRIPTION

5.1DATA ENTRY MODULE

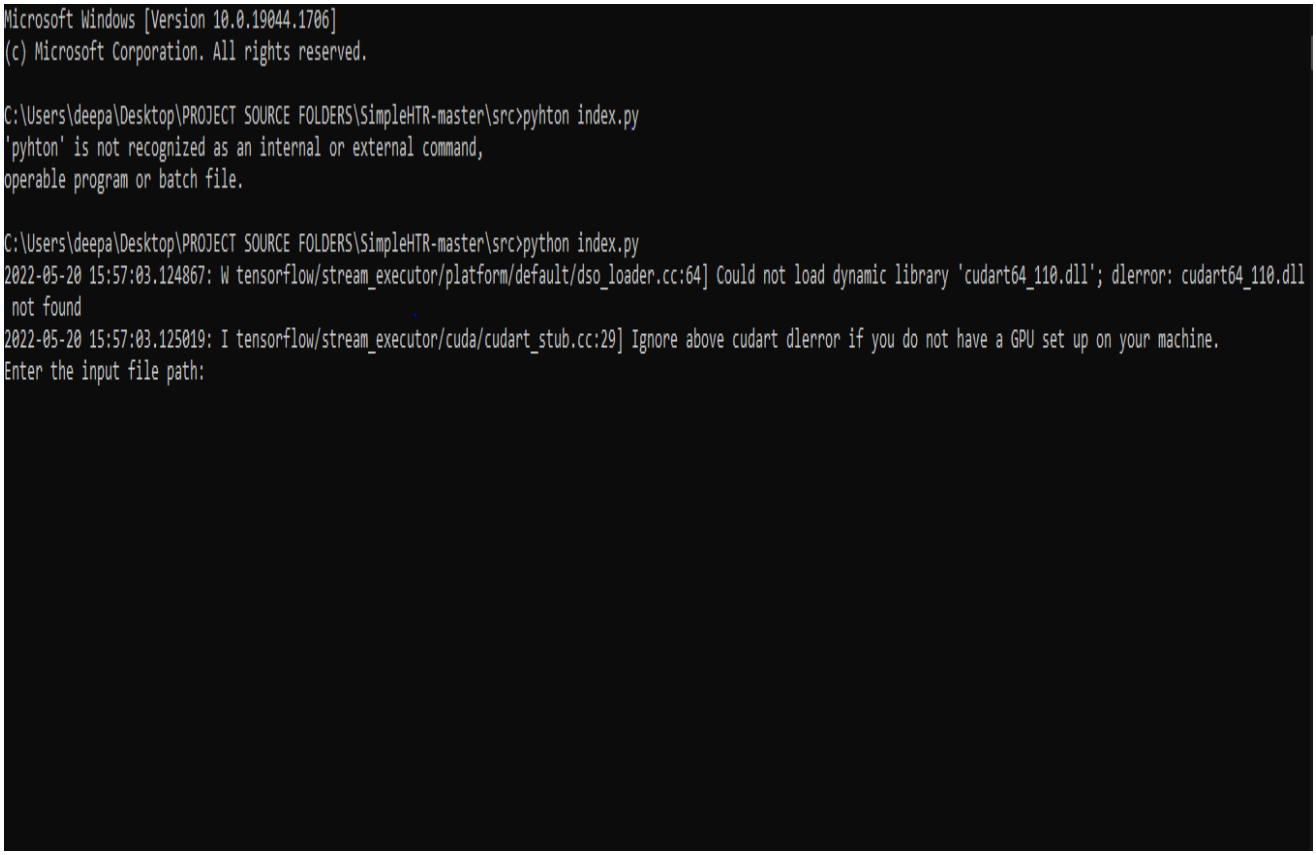

Figure 5.1Data Entry Module

- The student has to write the answer within the box so that the system finds it easy to scan the answers only inside the box to detect the handwritten answers.

- The staff can have the downloaded file anywhere in the system required they need to enter the path as like ../desktop/admin.filetype

5.2 RECOGNITION MODULE

Figure 5.1 Recognition Module

- The staff tries to enter the number of answers according to the questions paper and the correct answers.

5.2 ALGORITHMS

5.2 HANDWRITTEN TEXT RECOGNITION

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

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 5.2 shows an summary of our HTR system.

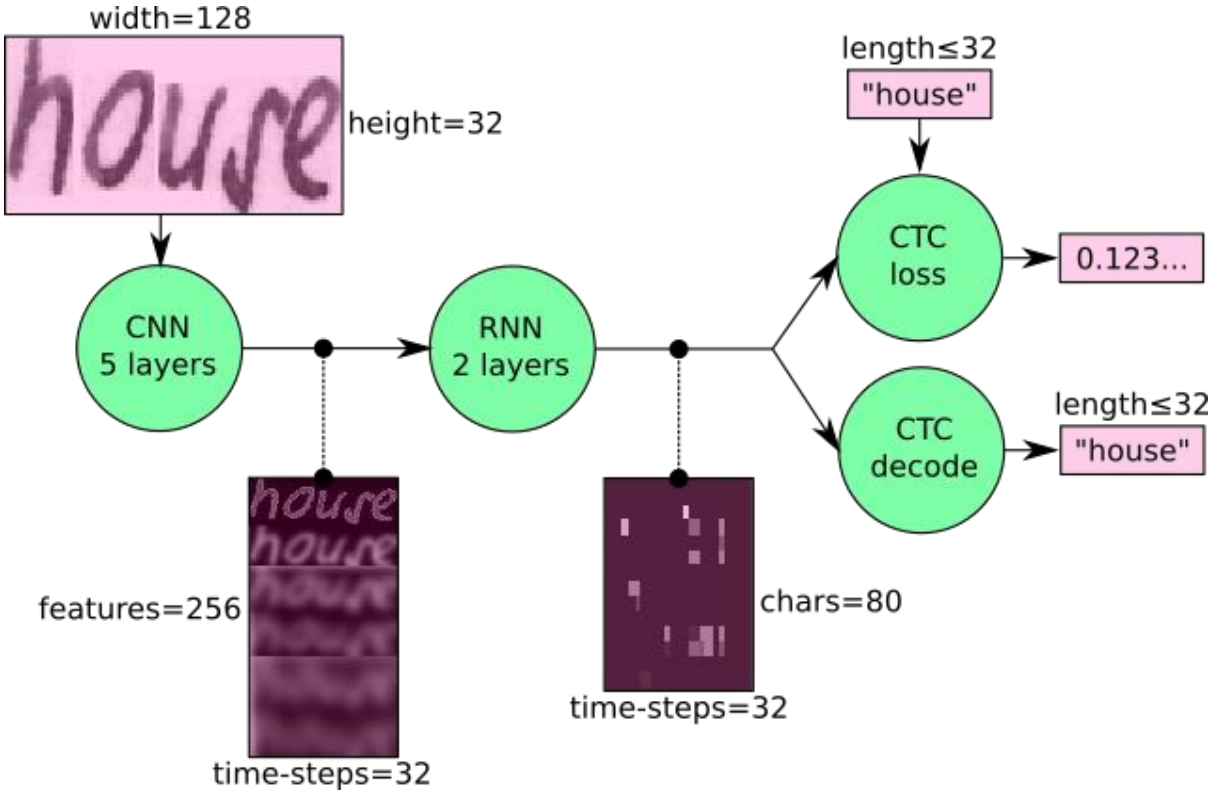

Figure 5.2: Overview of the NN operations (green) and the data flow through the NN (pink).

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

Eq. 1: The NN written as a function which maps a picture M to a personality sequence(c1,c2,…)

5.2.2CNN

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 within the first two layers and 3×3 within 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, in order that the output feature map (or sequence) encompasses a size of 32×256.

5.2.3RNN

RNN: the feature sequence contains 256 features per time-step, the RNN propagates relevant information through this sequence. the popular Long remembering (LSTM) implementation of RNNs is employed, The IAM dataset consists of 79 different characters, further one additional character is required for the CTC operation (CTC blank label), therefore there are 80 entries for every of the 32 time-steps.

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

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

CHAPTER 6

SYSTEM IMPLEMENTATION

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 SAMPLE CODING

```
import onlyRectangle as rec

import main

import numpy as np

import cv2

pls=[]

print("Enter the input file path:")

path=input()

print("Enter the no of ans:")

n=int(input())

print("Enter the answers:")

for i in range(0,n):

        pls.append(input())



imgs = rec.recognize(path)

print(len(imgs))
```

```
rec=[]

pro=[]

i=0

for img in imgs:

    i=i+1

    (thresh, img) = cv2.threshold(img, 160,255, cv2.THRESH_BINARY)

    cv2.imshow(str(i)+".",img)

    recognized, probability = main.predict(img)

    rec.append(recognized)

    pro.append(probability)

print(rec)

score=0

for i in range(0,len(rec)):

        if(str(rec[i][0]).lower()==pls[i].lower()):

                score=score+1

        else:

                print("wrong answer")

print(score)
```

6.2 Main.py

```
import argparse

import json

from typing import Tuple, List


import cv2

import edit_distance

from path import Path


from dataloader_iam import DataLoaderIAM, Batch

from model import Model, DecoderType

from preprocessor import Preprocessor



class FilePaths:

    """Filenames and paths to data."""

    fn_char_list = '../model/charList.txt'

    fn_summary = '../model/summary.json'
```

```
    fn_corpus = '../data/corpus.txt'




def get_img_height() -> int:
    """Fixed height for NN."""
    return 32




def get_img_size(line_mode: bool = False) -> Tuple[int, int]:
    """Height is fixed for NN, width is set according to training mode (single words or text lines)."""
    if line_mode:
        return 256, get_img_height()
    return 128, get_img_height()




def write_summary(char_error_rates: List[float], word_accuracies: List[float]) -> None:
    """Writes training summary file for NN."""
```

```
    with open(FilePaths.fn_summary, 'w') as f:
        json.dump({'charErrorRates': char_error_rates, 'wordAccuracies': word_accuracies}, f)


def char_list_from_file() -> List[str]:
    with open(FilePaths.fn_char_list) as f:
        return list(f.read())


def train(model: Model,
          loader: DataLoaderIAM,
          line_mode: bool,
          early_stopping: int = 25) -> None:
    """Trains NN."""
    epoch = 0  # number of training epochs since start
    summary_char_error_rates = []
    summary_word_accuracies = []
```

```
    preprocessor = Preprocessor(get_img_size(line_mode), data_augmentation=True, line_mode=line_mode)

    best_char_error_rate = float('inf')  # best validation character error rate

    no_improvement_since = 0  # number of epochs no improvement of character error rate occurred

    # stop training after this number of epochs without improvement

    while True:

        epoch += 1

        print('Epoch:', epoch)



        # train

        print('Train NN')

        loader.train_set()

        while loader.has_next():

            iter_info = loader.get_iterator_info()

            batch = loader.get_next()

            batch = preprocessor.process_batch(batch)

            loss = model.train_batch(batch)

            print(f'Epoch: {epoch} Batch: {iter_info[0]}/{iter_info[1]} Loss: {loss}')
```

```
        # validate
        char_error_rate, word_accuracy = validate(model, loader, line_mode)

        # write summary
        summary_char_error_rates.append(char_error_rate)
        summary_word_accuracies.append(word_accuracy)
        write_summary(summary_char_error_rates, summary_word_accuracies)

        # if best validation accuracy so far, save model parameters
        if char_error_rate < best_char_error_rate:
            print('Character error rate improved, save model')
            best_char_error_rate = char_error_rate
            no_improvement_since = 0
            model.save()
        else:
            print(f'Character error rate not improved, best so far: {char_error_rate * 100.0}%')
            no_improvement_since += 1
```

```
        # stop training if no more improvement in the last x epochs
        if no_improvement_since >= early_stopping:
            print(f'No more improvement since {early_stopping} epochs. Training stopped.')
            break


def validate(model: Model, loader: DataLoaderIAM, line_mode: bool) -> Tuple[float, float]:
    """Validates NN."""
    print('Validate NN')
    loader.validation_set()
    preprocessor = Preprocessor(get_img_size(line_mode), line_mode=line_mode)
    num_char_err = 0
    num_char_total = 0
    num_word_ok = 0
    num_word_total = 0
```

```
    while loader.has_next():
        iter_info = loader.get_iterator_info()
        print(f'Batch: {iter_info[0]} / {iter_info[1]}')
        batch = loader.get_next()
        batch = preprocessor.process_batch(batch)
        recognized, _ = model.infer_batch(batch)


        print('Ground truth -> Recognized')
        for i in range(len(recognized)):
            num_word_ok += 1 if batch.gt_texts[i] == recognized[i] else 0
            num_word_total += 1
            dist = editdistance.eval(recognized[i], batch.gt_texts[i])
            num_char_err += dist
            num_char_total += len(batch.gt_texts[i])
            print('[OK]' if dist == 0 else '[ERR:%d]' % dist, "" + batch.gt_texts[i] + "",
'->',
                  "" + recognized[i] + "")


    # print validation result
```

```
    char_error_rate = num_char_err / num_char_total

    word_accuracy = num_word_ok / num_word_total

    print(f'Character error rate: {char_error_rate * 100.0}%. Word accuracy: {word_accuracy * 100.0}%.')

    return char_error_rate, word_accuracy




def infer(model: Model, fn_img: Path) -> None:

    """Recognizes text in image provided by file path."""

    img = cv2.imread(fn_img, cv2.IMREAD_GRAYSCALE)

    assert img is not None


    preprocessor = Preprocessor(get_img_size(), dynamic_width=True, padding=16)

    img = preprocessor.process_img(img)


    batch = Batch([img], None, 1)

    recognized, probability = model.infer_batch(batch, True)

    print(f'Recognized: "{recognized[0]}"')
```

```
    print(f'Probability: {probability[0]}')



def parse_args() -> argparse.Namespace:

    """Parses arguments from the command line."""

    parser = argparse.ArgumentParser()


    parser.add_argument('--mode', choices=['train', 'validate', 'infer'], default='infer')

    parser.add_argument('--decoder', choices=['bestpath', 'beamsearch', 'wordbeamsearch'], default='wordbeamsearch')

    parser.add_argument('--batch_size', help='Batch size.', type=int, default=100)

    parser.add_argument('--data_dir', help='Directory containing IAM dataset.', type=Path, required=False)

    parser.add_argument('--fast', help='Load samples from LMDB.', action='store_true')

    parser.add_argument('--line_mode', help='Train to read text lines instead of single words.', action='store_true')

    parser.add_argument('--img_file', help='Image used for inference.', type=Path, default='../data/word.png')
```

```
    parser.add_argument('--early_stopping', help='Early stopping epochs.', type=int, default=25)

    parser.add_argument('--dump', help='Dump output of NN to CSV file(s).', action='store_true')


    return parser.parse_args()



def main():
    """Main function."""


    # parse arguments and set CTC decoder
    args = parse_args()
    decoder_mapping = {'bestpath': DecoderType.BestPath,
                       'beamsearch': DecoderType.BeamSearch,
                       'wordbeamsearch': DecoderType.WordBeamSearch}
    decoder_type = decoder_mapping[args.decoder]


    # train the model
```

```
    if args.mode == 'train':
        loader = DataLoaderIAM(args.data_dir, args.batch_size, fast=args.fast)

        # when in line mode, take care to have a whitespace in the char list
        char_list = loader.char_list
        if args.line_mode and ' ' not in char_list:
            char_list = [' '] + char_list

        # save characters and words
        with open(FilePaths.fn_char_list, 'w') as f:
            f.write(''.join(char_list))

        with open(FilePaths.fn_corpus, 'w') as f:
            f.write(' '.join(loader.train_words + loader.validation_words))

        model = Model(char_list, decoder_type)
        train(model, loader, line_mode=args.line_mode,
early_stopping=args.early_stopping)
```

```
    # evaluate it on the validation set
    elif args.mode == 'validate':
        loader = DataLoaderIAM(args.data_dir, args.batch_size, fast=args.fast)
        model = Model(char_list_from_file(), decoder_type, must_restore=True)
        validate(model, loader, args.line_mode)


    # infer text on test image
    elif args.mode == 'infer':
        model = Model(char_list_from_file(), decoder_type, must_restore=True, dump=args.dump)
        infer(model, args.img_file)
def inferPredict(model: Model, image) -> None:
    """Recognizes text in image provided by file path."""
    img = image
    assert img is not None


    preprocessor = Preprocessor(get_img_size(), dynamic_width=True, padding=16)
    img = preprocessor.process_img(img)
```

```
    batch = Batch([img], None, 1)
    recognized, probability = model.infer_batch(batch, True)
    print(f'Recognized: "{recognized[0]}"')
    print(f'Probability: {probability[0]}')
    del model
    return recognized, probability
```

main.py

```
def predict(image):
    """Main function."""

    # parse arguments and set CTC decoder
    args = parse_args()
    decoder_mapping = {'bestpath': DecoderType.BestPath,
                       'beamsearch': DecoderType.BeamSearch,
                       'wordbeamsearch': DecoderType.WordBeamSearch}
    decoder_type = decoder_mapping[args.decoder]
```

```
    # train the model
    if args.mode == 'train':
        loader = DataLoaderIAM(args.data_dir, args.batch_size, fast=args.fast)

        # when in line mode, take care to have a whitespace in the char list
        char_list = loader.char_list
        if args.line_mode and ' ' not in char_list:
            char_list = [' '] + char_list

        # save characters and words
        with open(FilePaths.fn_char_list, 'w') as f:
            f.write(''.join(char_list))

        with open(FilePaths.fn_corpus, 'w') as f:
            f.write(' '.join(loader.train_words + loader.validation_words))

        model = Model(char_list, decoder_type)
        train(model, loader, line_mode=args.line_mode,
early_stopping=args.early_stopping)
```

```
    # evaluate it on the validation set
    elif args.mode == 'validate':
        loader = DataLoaderIAM(args.data_dir, args.batch_size, fast=args.fast)
        model = Model(char_list_from_file(), decoder_type, must_restore=True)
        validate(model, loader, args.line_mode)


    # infer text on test image
    elif args.mode == 'infer':
        model = Model(char_list_from_file(), decoder_type, must_restore=True,
dump=args.dump)
        recognized, probability = inferPredict(model, image)
        return recognized, probability
```

6.3 onlyRectangle.py

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def getContours(imgOriginal,img, imgContour):
    results=[]
```

```
    contours, heirarchy = cv2.findContours(img, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
    print(heirarchy[0])
    for i in range(0,len(contours)):
        if(heirarchy[0][i][3]>=0):#if has a parent contour then don't execute
            continue
        area = cv2.contourArea(contours[i])
        if area > 1000:
##            cv2.drawContours(imgContour, contours[i], -1, (255, 0, 0), 5)
            peri = cv2.arcLength(contours[i], True)
            approx = cv2.approxPolyDP(contours[i], 0.2* peri, True)
            cv2.drawContours(imgContour, approx, -1, (255, 255, 0), 8)
            x, y, w, h = cv2.boundingRect(approx)
            cv2.rectangle(imgContour, (x, y), (x + w, y + h), (0, 255, 0), 5)
            cv2.drawContours(imgContour, approx, -1, (0, 0, 255), 9)
##            cv2.imshow('contour',imgContour)
            edges=len(approx)
            if edges<= 4:
                results.append(imgContour[y+5:y+h-5,x+5:x+w-5])
                ratio = w / h
    return results
def recognize(path):
            img=cv2.imread(path, cv2.IMREAD_GRAYSCALE)
            imgContour = img.copy()
            imgBlur = cv2.GaussianBlur(img, (15,15), 5)
##            cv2.imshow("blur",imgBlur)
            imgCanny = cv2.Canny(imgBlur, 21, 73)
##            cv2.imshow("canny",imgCanny)
            kernel = np.ones((7,7))
```

```
        imgDilate = cv2.dilate(imgCanny, kernel, iterations=1)
##          cv2.imshow("dialte",imgDilate)
        results=getContours(img,imgDilate, imgContour)
        return results
```

CHAPTER 7

SYSTEM TESTING

CHAPTER 7

SYSTEM TESTING

7.1 SYSTEM TESTING

System Testing could be a level of testing that validates the entire and fully integrated software package. the aim of a system test is to judge the end-to-end system specifications. Usually, the software is merely one element of a bigger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is truly a series of various tests whose sole purpose is to exercise the complete computer-based system.

7.2 UNIT TESTING

Unit testing, a testing technique using which individual modules are tested to see if there are any issues by the developer himself. it's concerned with functional correctness of the standalone modules.The main aim is to isolate each unit of the system to spot, analyze and fix the defects.In this system each modules were tested individually and output was verified.

7.3 INTEGRATION TESTING

Integration testing is that the testing of varied modules of the software under development together as a bunch. This determines whether or not they function together seamlessly as a part of the system or whole.If the system runs consequently no errors were found atlast.

7.4 TEST CASES

S.NO	SCENARIO	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	PASS/FAIL
1.	Enter the file path	User enters the document path	The correct path destination	Correct file path	PASS
2.	According to the question paper the staff enters the number of answers	Enters the number of questions	Correct number of answers	Type the number of answers	PASS
3.	Entering the correct answers relevant to the question	Enter the answers	Specifying the answers to be correct or not	System checks to the answers and	PASS

CHAPTER 8

CONCLUSION

CHAAPTER 8
CONCLUSION

8.1 CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION: We discussed a NN which is in a position to acknowledge text in images. The NN consists of 5 CNN and a couple of 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 a few important parts of the code were presented. Finally, hints to enhance the popularity accuracy got.

FUTURE ENHANCEMENT: This technique supports image file to as input therefore the future enhancement will e like pdf to image file transfer so pdf types even be supported by the system.
This project is an integration of already contained application so doesn't contain any UI .UI design part is made also as a enhancement.

APPENDICES

A.1 SAMPLE INPUT AND OUTPUT SCREEN SHOTS

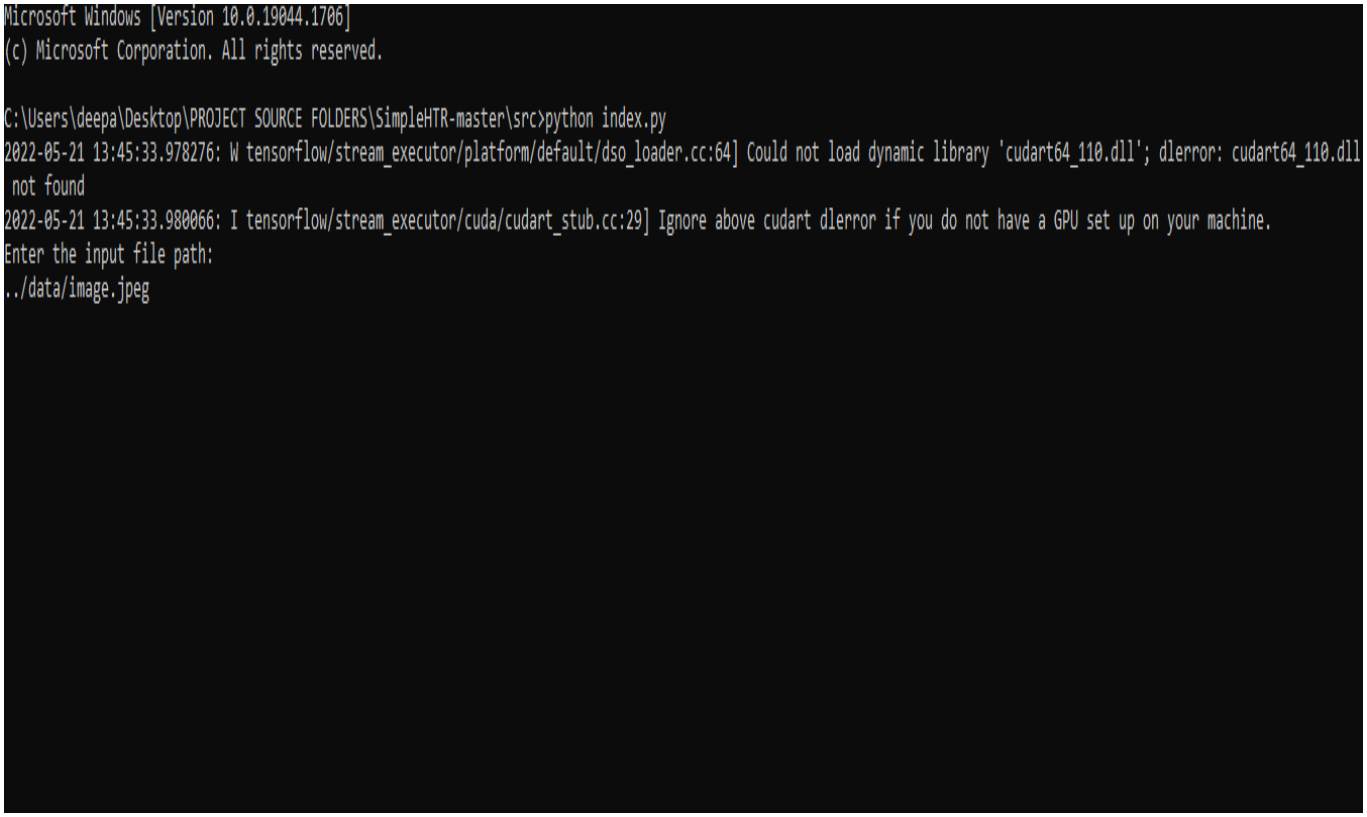

Figure A.1 path entering page

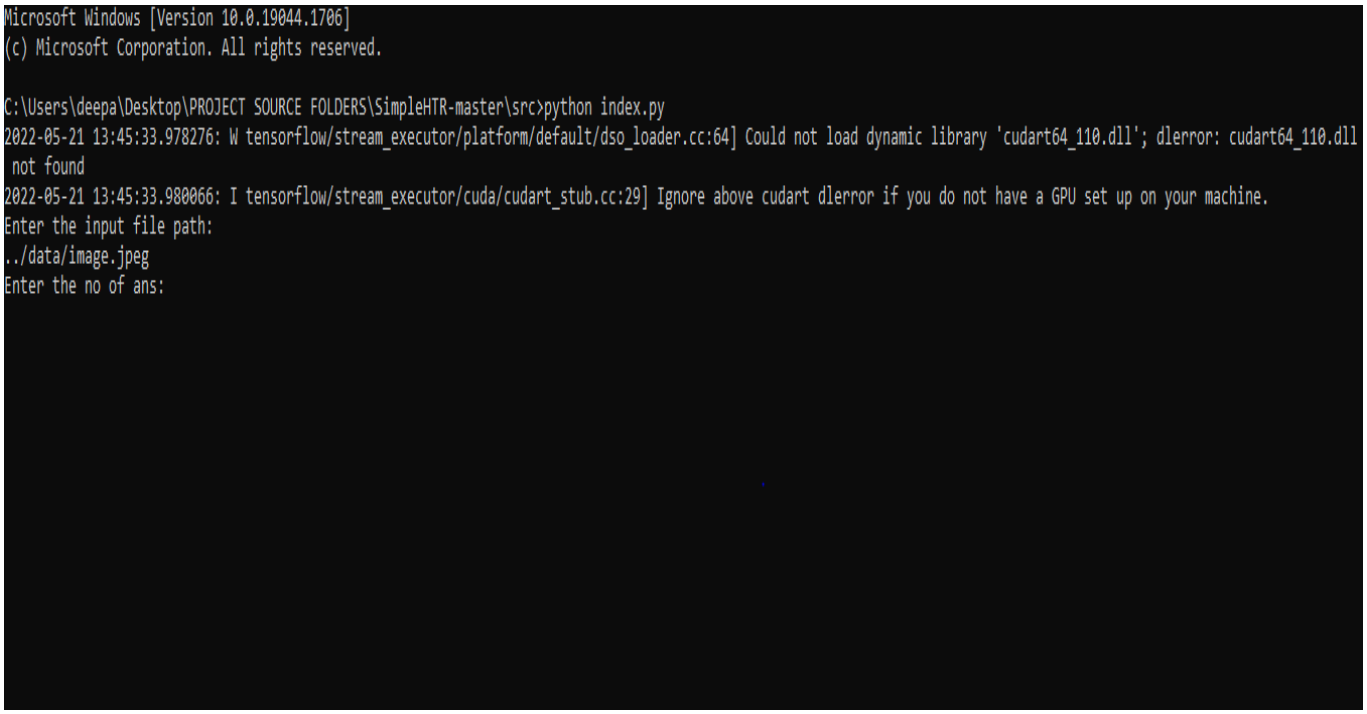

Figure A.1 answers entering page

```
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\deepa\Desktop\PROJECT SOURCE FOLDERS\SimpleHTR-master\src>python index.py
2022-05-21 13:45:33.978276: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll
 not found
2022-05-21 13:45:33.980066: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
Enter the input file path:
../data/image.jpeg
Enter the no of ans:
2
Enter the answers:
value
moral
```

Figure A.1 answer entering page

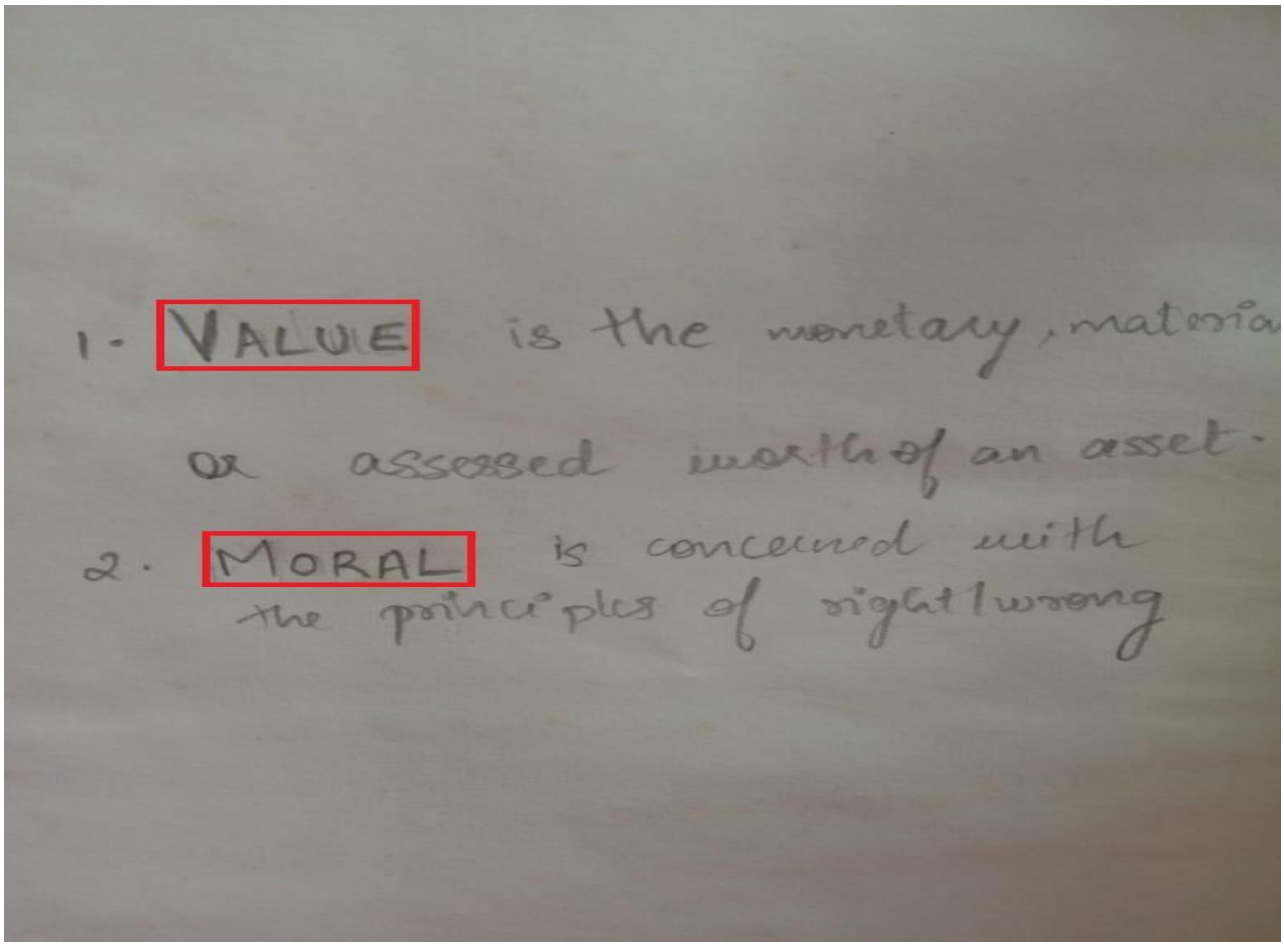

Figure A.1 answer paper

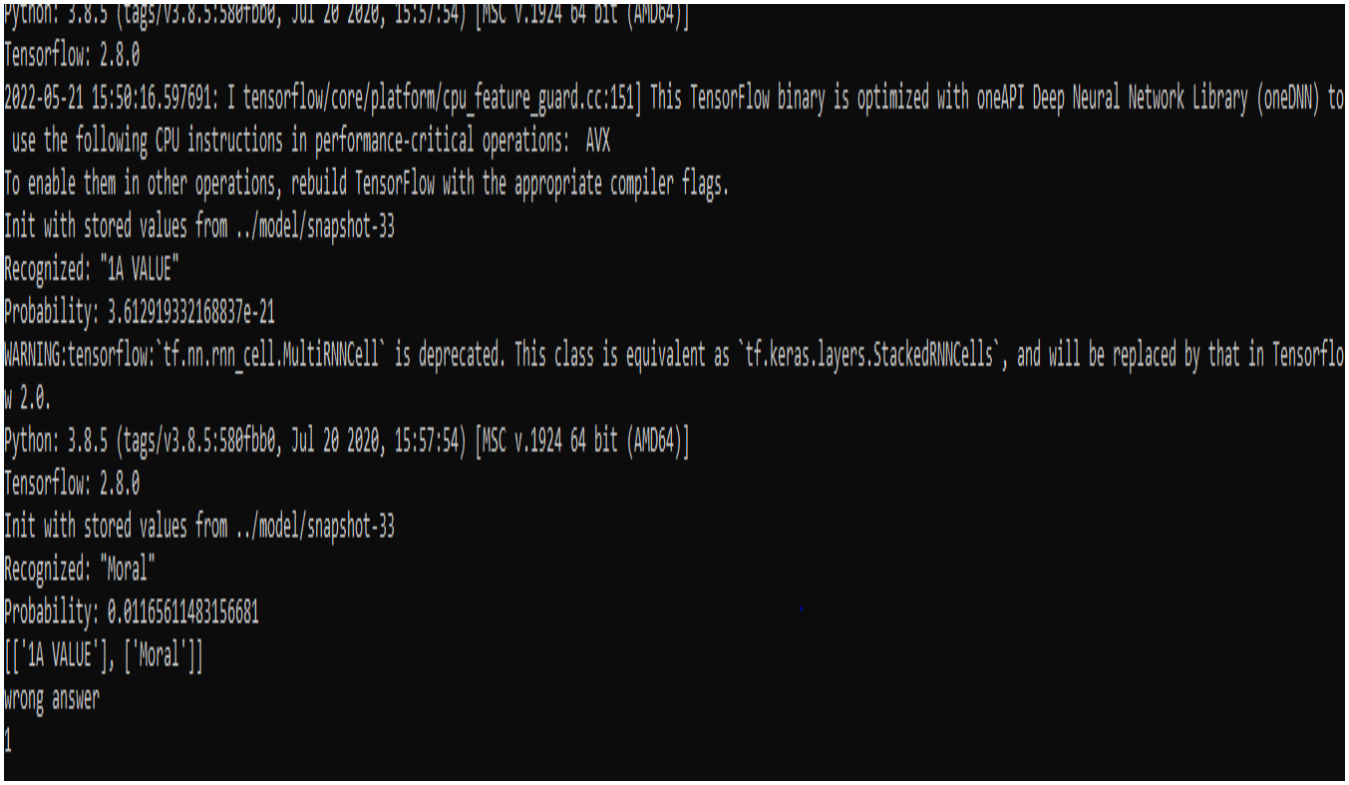

Figure A.1Recognition output page

REFERENCES

[1]Shilpa Mangesh Pande, Bineet Kumar Jha,Character Recognition System for Devanagari Script Using Machine Learning Approach, Conference on Information and Communication Technology (CICT'18), 2021

[2]Naragudem Sarika, Nageswararao Sirisala, Muni Sekhar Velpuru ,CNN based Optical Character Recognition and Applications, Proceedings of the Sixth International Conference on Inventive Computation Technologies, 2021

[3]Ayan Kumar Bhunia, Shuvozit Ghose, Amandeep Kumar, Pinaki Nath Chowdhury, Aneeshan Sain, Yi-Zhe,Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021

[5]Daniyar Nurseitov, Kairat Bostanbekov, Maksat Kanatov, Anel Alimova, Abdelrahman Abdallah, Galymzhan Abdimanap, Classification of Handwritten Names of Cities and Handwritten Text Recognition using Various Deep Learning Models, Advances in Science, Technology and Engineering Systems. 5, 934-943 ,2021

[6]A Scalable Handwritten Text Recognition System, R. Reeve Ingle, Yasuhisa Fujii, Thomas Deselaers, Jonathan Baccash, Ashok C. Popat, 9 International Conference on Document Analysis and Recognition (ICDAR),2019

[7]Tianwei Wang,Yuanzhi Zhu, Lianwen Jin, Canjie Luo, Xiaoxue Chen,Decoupled Attention Network for Text Recognition, School of Electronic and Information Engineering, South China University of Technology 2Lenovo Research,2019

[8]Rajib Ghosh, Chinmaya Panda,Handwritten Text Recognition in Bank Cheques, Conference on Information and Communication Technology (CICT'18),2018

[9] Zhang, Y.; Nie, S.; Liu, W.; Xu, X.; Zhang, D.; and Shen, H. T. 2019. Sequence-to-sequence domain adaptation network for robust text image recognition. In IEEE Conference on Computer Vision and Pattern Recognition, 2740–2749

[10] Zhan, F., and Lu, S. 2019. Esir: End-to-end scene text recognition via iterative image rectification. In IEEE Conference on Computer Vision and Pattern Recognition, 2059–2068

[11] Xie, Z.; Huang, Y.; Zhu, Y.; Jin, L.; and Xie, L. 2019. Aggregation cross-entropy for sequence recognition. In IEEE Conference on Computer Vision and Pattern Recognition, 6538–6547. Yang, X.; He, D.; Zhou, Z.; Kifer, D.; and Giles, C. L. 2017.

[12] Learning to read irregular text with attention mechanisms. In International Joint Conference on Artificial Intelligence, 3280–3286. Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. CoRR abs/1212.5701

[13] Sueiras, J.; Ruiz, V.; Sanchez, A.; and Velez, J. F. 2018. Offline continuous handwriting recognition using sequence to sequence neural networks. Neurocomputing 289:119–128

[14] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In Annual Conference on Neural Information Processing Systems, 5998–6008.

[15] Shi, B.; Yang, M.; Wang, X.; Lyu, P.; Yao, C.; and Bai, X. 2018. ASTER: An attentional scene text recognizer with flexible rectification. IEEE Trans. Pattern Anal. Mach. Intell.

[16] Sueiras, J.; Ruiz, V.; Sanchez, A.; and Velez, J. F. 2018. Offline continuous handwriting recognition using sequence to sequence neural networks. Neurocomputing 289:119–128.

[17] Shi, B.; Yang, M.; Wang, X.; Lyu, P.; Yao, C.; and Bai, X. 2018. ASTER: An attentional scene text recognizer with flexible rectification. IEEE Trans. Pattern Anal. Mach. Intell.

[18] Liao, M.; Zhang, J.; Wan, Z.; Xie, F.; Liang, J.; Lyu, P.; Yao, C.; and Bai, X. 2019. Scene text recognition from twodimensional perspective. In AAAI Conference on Artificial Intelligence, 8714–8721.