Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 8, July 2021: 1874 - 1890

Deep Learning for Handwritten Text Recognition (ConvNet & RNN)

Rohini G. Khalkar[1], Adarsh Singh Dikhit[2], Anirudh Goel[3], Manisha Gupta[4], Sheetal S. Patil[5]

[1,2,3,4,5] Department of Computer Engineering, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, India-411043

ABSTRACT

In computing, Handwritten Text Recognition refers to the ability and method of a computer to recognise and understand legible handwriting acquired from a variety of sources, including photos, scanned documents, and other sources. Machine learning techniques like handwritten text recognition are used to create patterns. To achieve pattern realisation, input or objects are organised or categorised into a small number of groups or categories from a vast number of possible options.Historically, handwriting identification methods depended on handmade characteristics and a significant quantity of preexisting knowledge.

Training an Handwritten Text Recognition system utilising the standard methods and a varied collection of rules and criteria is a difficult job. Numerous research on handwriting recognition have been conducted in recent years, with a special focus on deep learning methods that have produced breakthrough results. Despite this, the increasing amount of handwritten data and the new strides in computing power are very much need to increase the actual accuracy achieved in properly recognising written text, which necessitates further study. Consonant with the fact that convolutional neural networks (CONVNETs) algorithms are extremely efficient at extracting structure from images, they are also extremely capable of identifying handwritten textual characters/words in ways that facilitate automatic recognition of unique characteristics. As a result, CONVNETs algorithms are the most appropriate method for addressing handwritten text recognition challenges.

This technique will be used to recognise text in a number of different forms. Handwriting has developed into a more sophisticated art form, as shown by the existence of a wide variety of handwritten textual characters, including numbers, characters, scripts, and symbols, in both English and other languages.

Keywords: *RNN, CTC, LSTM, SGD, ReLU, BLSTM, Tensor Flow, SoftMax*

INTRODUCTION

A Handwritten Text Recognition system is primarily concerned with feature extraction and feature discrimination/classification (based on patterns). Handwritten Text Recognition is a much-needed technique today. Prior to the effective application of this technology, we depended on handwriting texts, which is prone to mistake. Physical data is notoriously inefficient to store and access. To keep the data organized properly, manual effort is needed. Throughout history, significant data loss has occurred because of conventional data storage methods. Modern technology enables individuals to store data on computers, which facilitates data storage, organization, and access.

It is much simpler to save and retrieve data that was previously stored using Handwritten Text Recognition software. Additionally, it strengthens the data's security. Google Lens is an example of such software for handwritten text recognition. Our project's objective is to develop a deep learning-based application capable of recognizing handwriting. We believe that by addressing our issue via the lens of CONVNET, we may get a higher degree of accuracy.

OBJECTIVE:

Handwritten Textual Identification is a subset of pattern identification and is denoted by the phrase "textual recognition." In pattern recognition, the goal is to organise or categorise data or objects into one of a wide number of groups or categories, which may then be used to make decisions. Handwriting As described by the scientific community, textual recognition is the process of converting a spatially marked expression/language to its symbolic equivalent. Scripts are made up of a collection of symbols known as characters or letters, which are organised according to a set of fundamental forms.

The objective of handwriting is to accurately recognize input letters or images, which are subsequently evaluated by a variety of automatic process systems. The method used to recognize various types of writing is quite similar. Handwriting has advanced in sophistication, as shown by the presence of many types of handwritten characters, including digits, numerals, cursive script, symbolic expressions, and characters in English and alternate languages. Mechanized identification of handwritten text could be utmost beneficial in a variety of applications that require humongous quantities of handwritten data to be processed, in the likes of addresses and postcode recognition on envelopes, amount interpretation on banking instruments,analysis of various documents, and signature verification. Thus, a machine must be capable of reading documents or data to facilitate document processing.

CONVNET image classifications analyses and categorises an input image (E.g., Alligator, Cat Family, Tiger, Lion). The entering image is interpreted by the computer as a collection of pixels.. For example, each picture is comprised of a 8 x 8 x 3 matrix of RGB values (3 denotes RGB values) and a 8 x 8 x 1 matrix of image having grayscale values, here 3 and 1 denote the number of colour values required to represent each image pixel, respectively.

Below Mentioned layers are often seen in ConvNets:
1. **Convolution**
2. **Striding**
3. **non-Linearity**
4.**Pooling Layer**

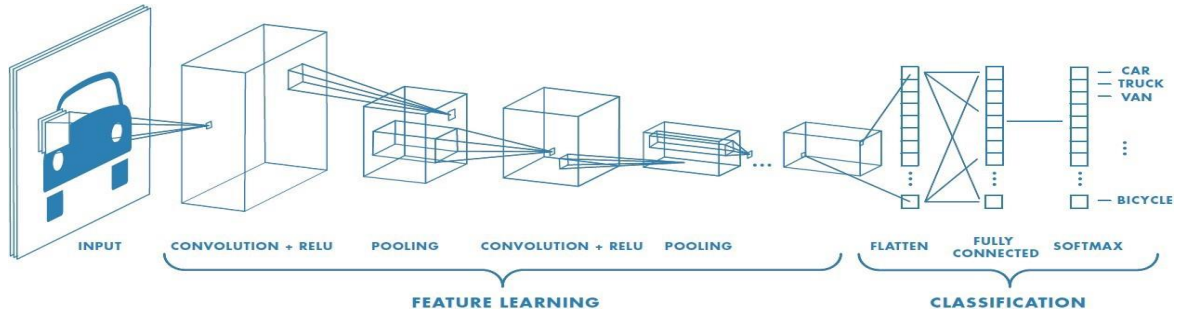


Fig 1: Architecture of Convolutional Neural Networks

LITERATURE REVIEW:

We chose the MNIST [1] dataset since it has been utilised in a number of important accomplishments. Even before the introduction of DL, handwritten text recognition was achievable, but the accuracies achieved were low, or the datasets were small, as Line Evil said [2]. This page covers the many uses of optical character recognition (OCR), including voice recognition and a variety of other tasks. Classification of the MNIST dataset, which consists of a collection of numbers, is a frequently encountered machine learning task. The paper "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis" by Simard, Steinkraus, and Platt is an excellent introduction on how to utilise convoluted neural networks (ConvNets). Pham et al. trained a bidirectional RNN incorporating LSTM cells for word recognition using a two-layer ConvNet [3].

Schmid and Graves Uber's approach, which uses a multidimensional RNN structure, is the best implementation, in our view. [4] M.J. Castro-'Handwritten Bleda's Text Recognition' also dealt with datasets including slanted words and corrected them during pre-processing. [5] Teddy Surya and Ahmad Fakhrur used the EMNIST dataset to build a Deep ANNmodel with two Encoding layers and one SoftMax layer for English handwriting recognition. Their accuracy was much higher than that of the previously proposed patterned and feedforward network artificial neural network ANN[6]. We achieved an accuracy of over 88.5 percent by using ConvNets from the Keras[8] toolbox.

The ConvNet model has been extensively utilised in recent years to recognise handwritten digits from the MNIST benchmark collection. Certain studies have reported an accuracy of up to 95 or 96 percent for HDR [8]. An ensemble model was created by combining multiple ConvNet models. The recognition experiment utilised MNIST digits, and the accuracy was claimed to be 99.73 percent [9]. after, this "7-net committee" experiment was incorporated to a "35-net committee," resulting in a 99.77 percent recognition accuracy for the identical

MNIST dataset. Niu and Suen obtained an amazing 99.81 percent identification accuracy for the MNIST digit recognition experiment by combining the SVM's structural risk reduction capabilities with the ConvNet's deep information extraction capability [1].

The bend directional feature maps for in-air recognition of handwritten Chinese characters were explored using ConvNet [9]. Alvear-Sandoval et al. recently published a paper describing how they got a 0.19 percent error rate (loss function) for MNIST by developing several ensembles of DL neural networks. But, it has been found that the MNIST dataset images' steep identification accuracy is only achieved via ensemble methods. While ensemble methods enhance classification accuracy, they entail increased testing complexity and computational cost in real-world applications.

SYSTEM DESIGN

Handwritten Text Recognition (HTR) systems, as shown in Figure 1, make use of scanned images of handwritten text. We will train a Neural Network (NN) using the word images from the IAM dataset. Since the input layer and the following hidden layers are often kept short for word images, neural network training on the GPU is preferred always and maybe TPUs. ATF is the very bare minimum need for HTR deployment.

The recognition technique we are proposing is based on a ConvNets with an final layer that is feature-mapped. The model that we will develop will classify the input into ten categories using ConvNet while also classifying the a numeral. Similarly, when an character is classified, the same DL model assigns the character to one of twelve groups. The next subsections will discuss in detail the different phases of our proposed paradigm.

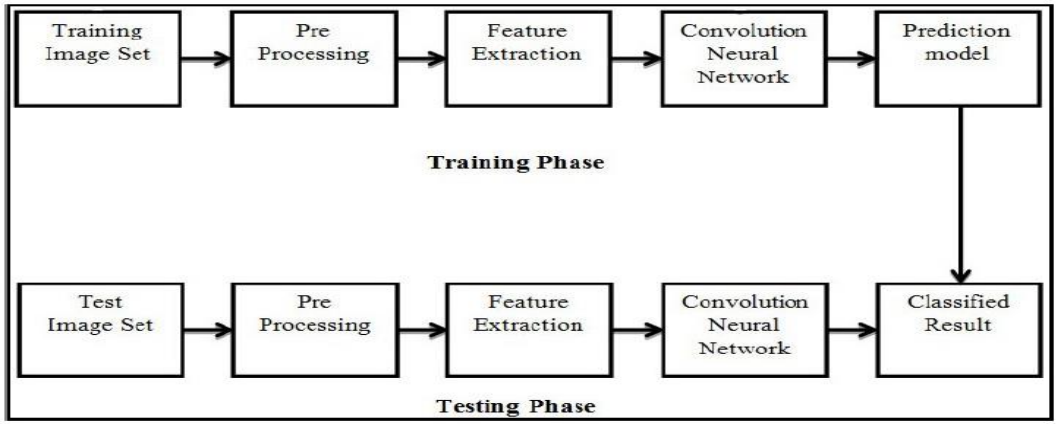


Fig 2: Schematic representation of a suggested method for classifying handwritten characters

Process Models are assemblages of interrelated processes. Thus, the proposed model is a depiction of a process at the top level. Due to the fact that the proposed model resides at the top level, its instantiation is a process to watch out for. Due to the fact that the same process model is used to create multiple applications, it has numerous instantiations. Unlike to the process which is what really happens, the propsed model may be used to prescribe how things must/should/could be done. The proposed model is an educated guess as to how the process will manifest itself. That the process will include and what will be determined during the system's actual development.The purpose of a proposed model is to:

A. Detailed:

1. Maintain a log of events that occur throughout a process.
2. Put yourself in the shoes of an external observer who analyses how a process is carried out and determines what adjustments should be made to improve its effectiveness or efficiency.

B. Normative:

1. Specify the processes to be followed and the manner in which, might be carried out.
2. Established rules, standards, and behavioural patterns that, when followed, will result in the process functioning well. They may range from rigidly enforcing the regulations to offering individualised guidance.

C. Descriptive:

1. Justify methods.
2. Investigate and evaluate different possible paths of action using logical reasoning.
3. Establish a clear link between processes and the model's required criteria.
4. Specifies data extraction points in advance for reporting purposes.

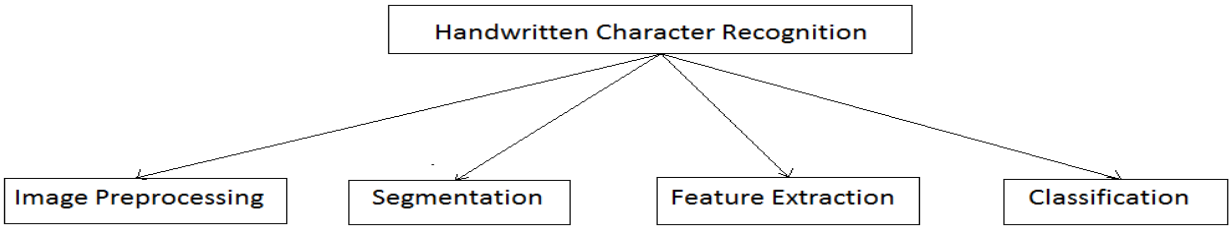


Fig 3: Break up of model

1. **Preparation of the Image**

Preprocessing of the image includes inversion, grayscale inversion, and image processing and especially thinning.

2. **Classification**

Segmentation by using Image Processing techniques for images is done after preprocessing. This is done by following the instructions below:

1. Remove any impediments
2.Divide the text into rows to shorten it.
3.Recategorize the rows (lines) according to their content.

4.Breaking the word into individual letters.

3. Feature Extraction

Following character segmentation, we spawn binary glyphs and utilise the sum in each row and column value as a feature.

4. Categorization

We will train and test the ANN in this phase to find the right answer.

Training the Model:

- After splitting the IAM dataset into training and testing sets.
- For Training set are using 7850 images.
- For Validation are using 876 images.
- we will load and utilise a portion of it.
- To begin, we want to create a model architecture that incorporates ConvNets, RNNs, and other desirable layers, all of which will be implemented using Python's deep learning modules.
- Then, we'll do any necessary pre-processing on the dataset like image inverting, Gray scale and image thinning.
- The next section is segmentation, which includes removing the border, dividing the text toward individual rows, dividing the lines inputted into individual words, and segmenting the words into individual letters.

1. Sample pre-process

```python
def preprocessor(img, imgSize, enhance=False, dataAugmentation=False):
    "put img into target img of size imgSize, transpose for TF and normalize gray-values"

    # there are damaged files in IAM dataset - just use black image instead
    if img is None:
        img = np.zeros([imgSize[1], imgSize[0]]) # (64,800)
        print("Image None!")

    # increase dataset size by applying random stretches to the images
    if dataAugmentation:
        stretch = (random.random() - 0.5)  # -0.5 .. +0.5
        wStretched = max(int(img.shape[1] * (1 + stretch)), 1)  # random width, but at least 1
        img = cv2.resize(img, (wStretched, img.shape[0]))  # stretch horizontally by factor 0.5 .. 1.5

    if enhance: # only if the line text has low contrast and line width is thin
        # increase contrast
        pxmin = np.min(img)
        pxmax = np.max(img)
        imgContrast = (img - pxmin) / (pxmax - pxmin) * 255
        # increase line width (optional)
        kernel = np.ones((3, 3), np.uint8)
        img = cv2.erode(imgContrast, kernel, iterations=1) # increase linewidth

    # create target image and copy sample image into it
    (wt, ht) = imgSize
    (h, w) = img.shape
    fx = w / wt
    fy = h / ht
    f = max(fx, fy)
    newSize = (max(min(wt, int(w / f)), 1),
               max(min(ht, int(h / f)), 1))  # scale according to f (result at least 1 and at most wt or ht)
    img = cv2.resize(img, newSize, interpolation=cv2.INTER_CUBIC) # INTER_CUBIC interpolation best approximate the pixels image
    target = np.ones([ht, wt]) * 255  # shape=(64,800)
    target[0:newSize[1], 0:newSize[0]] = img

    # transpose for TF
    img = cv2.transpose(target)

    return img
```

```python
def wer(r, h):

    # initialisation
    d = np.zeros((len(r)+1)*(len(h)+1), dtype=np.uint8)
    d = d.reshape((len(r)+1, len(h)+1))
    for i in range(len(r)+1):
        for j in range(len(h)+1):
            if i == 0:
                d[0][j] = j
            elif j == 0:
                d[i][0] = i

    # computation
    for i in range(1, len(r)+1):
        for j in range(1, len(h)+1):
            if r[i-1] == h[j-1]:
                d[i][j] = d[i-1][j-1]
            else:
                substitution = d[i-1][j-1] + 1
                insertion = d[i][j-1] + 1
                deletion = d[i-1][j] + 1
                d[i][j] = min(substitution, insertion, deletion)
    return d[len(r)][len(h)]
```

2. Apply ConvNet

```python
def setupCNN(self):
    """ Create CNN layers and return output of these layers """

    cnnIn4d = tf.expand_dims(input=self.inputImgs, axis=3)

    # First Layer: Conv (5x5) + Pool (2x2) - Output size: 400 x 32 x 64
    with tf.name_scope('Conv_Pool_1'):
        kernel = tf.Variable(
            tf.truncated_normal([5, 5, 1, 64], stddev=0.1))
        conv = tf.nn.conv2d(
            cnnIn4d, kernel, padding='SAME', strides=(1, 1, 1, 1))
        learelu = tf.nn.leaky_relu(conv, alpha=0.01)
        pool = tf.nn.max_pool(learelu, (1, 2, 2, 1), (1, 2, 2, 1), 'VALID')

    # Second Layer: Conv (5x5) + Pool (1x2) - Output size: 400 x 16 x 128
    with tf.name_scope('Conv_Pool_2'):
        kernel = tf.Variable(tf.truncated_normal(
            [5, 5, 64, 128], stddev=0.1))
        conv = tf.nn.conv2d(
            pool, kernel, padding='SAME', strides=(1, 1, 1, 1))
        learelu = tf.nn.leaky_relu(conv, alpha=0.01)
        pool = tf.nn.max_pool(learelu, (1, 1, 2, 1), (1, 1, 2, 1), 'VALID')

    # Third Layer: Conv (3x3) + Pool (2x2) + Simple Batch Norm - Output size: 200 x 8 x 128
    with tf.name_scope('Conv_Pool_BN_3'):
        kernel = tf.Variable(tf.truncated_normal(
            [3, 3, 128, 128], stddev=0.1))
        conv = tf.nn.conv2d(
            pool, kernel, padding='SAME', strides=(1, 1, 1, 1))
        mean, variance = tf.nn.moments(conv, axes=[0])
        batch_norm = tf.nn.batch_normalization(
            conv, mean, variance, offset=None, scale=None, variance_epsilon=0.001)
        learelu = tf.nn.leaky_relu(batch_norm, alpha=0.01)
        pool = tf.nn.max_pool(learelu, (1, 2, 2, 1), (1, 2, 2, 1), 'VALID')

    # Fourth Layer: Conv (3x3) - Output size: 200 x 8 x 256
    with tf.name_scope('Conv_4'):
        kernel = tf.Variable(tf.truncated_normal(
            [3, 3, 128, 256], stddev=0.1))
        conv = tf.nn.conv2d(
            pool, kernel, padding='SAME', strides=(1, 1, 1, 1))
        learelu = tf.nn.leaky_relu(conv, alpha=0.01)

    # Fifth Layer: Conv (3x3) + Pool(2x2) - Output size: 100 x 4 x 256
    with tf.name_scope('Conv_Pool_5'):
        kernel = tf.Variable(tf.truncated_normal(
            [3, 3, 256, 256], stddev=0.1))
        conv = tf.nn.conv2d(
            learelu, kernel, padding='SAME', strides=(1, 1, 1, 1))
        learelu = tf.nn.leaky_relu(conv, alpha=0.01)
        pool = tf.nn.max_pool(learelu, (1, 2, 2, 1), (1, 2, 2, 1), 'VALID')

    # Sixth Layer: Conv (3x3) + Pool(1x2) + Simple Batch Norm - Output size: 100 x 2 x 512
    with tf.name_scope('Conv_Pool_BN_6'):
        kernel = tf.Variable(tf.truncated_normal(
            [3, 3, 256, 512], stddev=0.1))
        conv = tf.nn.conv2d(
            pool, kernel, padding='SAME', strides=(1, 1, 1, 1))
        mean, variance = tf.nn.moments(conv, axes=[0])
        batch_norm = tf.nn.batch_normalization(
            conv, mean, variance, offset=None, scale=None, variance_epsilon=0.001)
        learelu = tf.nn.leaky_relu(batch_norm, alpha=0.01)
        pool = tf.nn.max_pool(learelu, (1, 1, 2, 1), (1, 1, 2, 1), 'VALID')


    # Seventh Layer: Conv (3x3) + Pool (1x2) - Output size: 100 x 1 x 512
    with tf.name_scope('Conv_Pool_7'):
        kernel = tf.Variable(tf.truncated_normal(
            [3, 3, 512, 512], stddev=0.1))
        conv = tf.nn.conv2d(
            pool, kernel, padding='SAME', strides=(1, 1, 1, 1))
        learelu = tf.nn.leaky_relu(conv, alpha=0.01)
        pool = tf.nn.max_pool(learelu, (1, 1, 2, 1), (1, 1, 2, 1), 'VALID')

        self.cnnOut4d = pool
```

3. Apply RNN with LSTM

```
def setupRNN(self):
    """ Create RNN layers and return output of these layers """
    # Collapse layer to remove dimension 100 x 1 x 512 --> 100 x 512 on axis=2
    rnnIn3d = tf.squeeze(self.cnnOut4d, axis=[2])

    # 2 layers of LSTM cell used to build RNN
    numHidden = 512
    cells = [tf.contrib.rnn.LSTMCell(
        num_units=numHidden, state_is_tuple=True, name='basic_lstm_cell') for _ in range(2)]
    stacked = tf.contrib.rnn.MultiRNNCell(cells, state_is_tuple=True)
    # Bi-directional RNN
    # BxTxF -> BxTx2H
    ((forward, backward), _) = tf.nn.bidirectional_dynamic_rnn(
        cell_fw=stacked, cell_bw=stacked, inputs=rnnIn3d, dtype=rnnIn3d.dtype)

    # BxTxH + BxTxH -> BxTx2H -> BxTx1X2H
    concat = tf.expand_dims(tf.concat([forward, backward], 2), 2)

    # Project output to chars (including blank): BxTx1x2H -> BxTx1xC -> BxTxC
    kernel = tf.Variable(tf.truncated_normal(
        [1, 1, numHidden * 2, len(self.charList) + 1], stddev=0.1))
    self.rnnOut3d = tf.squeeze(tf.nn.atrous_conv2d(value=concat, filters=kernel, rate=1, padding='SAME'), axis=[2
```

4. Apply TensorFlow GPU

```
def setupTF(self):
    """ Initialize TensorFlow """
    print('Python: ' + sys.version)
    print('Tensorflow: ' + tf.__version__)
    sess = tf.Session()  # Tensorflow session
    saver = tf.train.Saver(max_to_keep=3)  # Saver saves model to file
    modelDir = '../model/'
    latestSnapshot = tf.train.latest_checkpoint(modelDir)  # Is there a saved model?
    # If model must be restored (for inference), there must be a snapshot
    if self.mustRestore and not latestSnapshot:
        raise Exception('No saved model found in: ' + modelDir)
    # Load saved model if available
    if latestSnapshot:
        print('Init with stored values from ' + latestSnapshot)
        saver.restore(sess, latestSnapshot)
    else:
        print('Init with new values')
        sess.run(tf.global_variables_initializer())

    return (sess, saver)
```

5. Setup Batch

```
def trainBatch(self, batch, batchNum):
    """ Feed a batch into the NN to train it """
    sparse = self.toSpare(batch.gtTexts)
    rate = 0.001 # if you use the pretrained model to continue train

    evalList = [self.merge, self.optimizer, self.loss]
    feedDict = {self.inputImgs: batch.imgs, self.gtTexts: sparse, self.seqLen: [Model.maxTextLen] * Model.batchSize, self.learningR
    (loss_summary, _, lossVal) = self.sess.run(evalList, feedDict)
    # Tensorboard: Add loss_summary to writer
    self.writer.add_summary(loss_summary, batchNum)
    self.batchesTrained += 1
    return lossVal
```

Model Verification:

Due to the fact that the proposed models are trained in previous rounds won't be verified and tested on fresh data, we will get a range of accuracies for each model and will then be able to

select the most accurate model. This method enables us to now finish the development and implementation of our HTR solution for end clients.

```python
import os
from datetime import datetime
from flask import Flask, request, render_template, send_from_directory

from main import infer_by_web


app = Flask(__name__)

APP_ROOT = os.path.dirname(os.path.abspath('model/trainmodel'))

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/about")
def about():
    return render_template("about.html")


@app.route("/upload_page", methods=["GET"])
def upload_page():
    return render_template("upload.html")


@app.route("/upload", methods=["POST"])
def upload():
    # folder_name = request.form['uploads']
    target = os.path.join(APP_ROOT, 'F:/Final_model/src/static/')
    print(target)
    if not os.path.isdir(target):
        os.mkdir(target)
    print(request.files.getlist("file"))
    option = request.form.get('optionsPrediction')
    print("Selected Option:: {}".format(option))
    for upload in request.files.getlist("file"):
        print(upload)
        print("{} is the file name".format(upload.filename))
        filename = upload.filename
        # This is to verify files are supported
        ext = os.path.splitext(filename)[1]
        if (ext == ".jpg") or (ext == ".png"):
            print("File supported moving on...")
        else:
            render_template("Error.html", message="Files uploaded are not supported...")
        savefname = datetime.now().strftime('%Y-%m-%d_%H_%M_%S') + "."+ext
        destination = "/".join([target, savefname])
        print("Accept incoming file:", filename)
        print("Save it to:", destination)
        upload.save(destination)
        result = predict_image(destination, option)
        print("Prediction: ", result)
    # return send_from_directory("images", filename, as_attachment=True)
    return render_template("complete.html", image_name=savefname, result=result)


def predict_image(path, type):
    print(path)
    return infer_by_web(path, type)


if __name__ == "__main__":
    app.run(port=5000, debug=True)
```

```
Validation character error rate of saved model: 8.654728%
```

System Architecture:

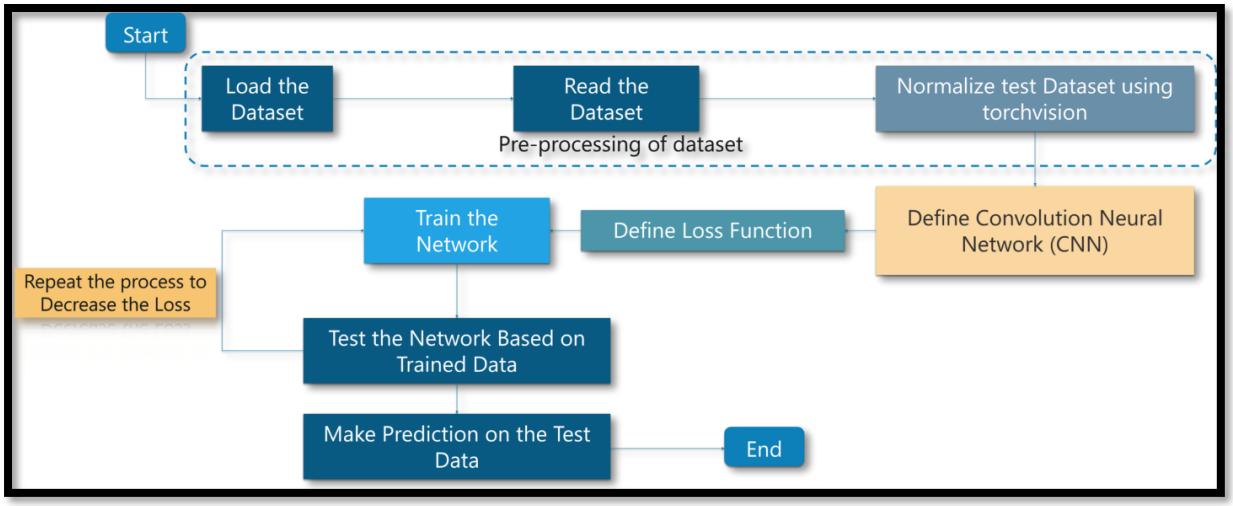


Fig 4: Diagram of the Modelling Process

The suggested Model Architecture entails converting and processing the input and then developing it into the ConvNet, that is then fed in the layers of RNN network, which finalizes, then produces the output.

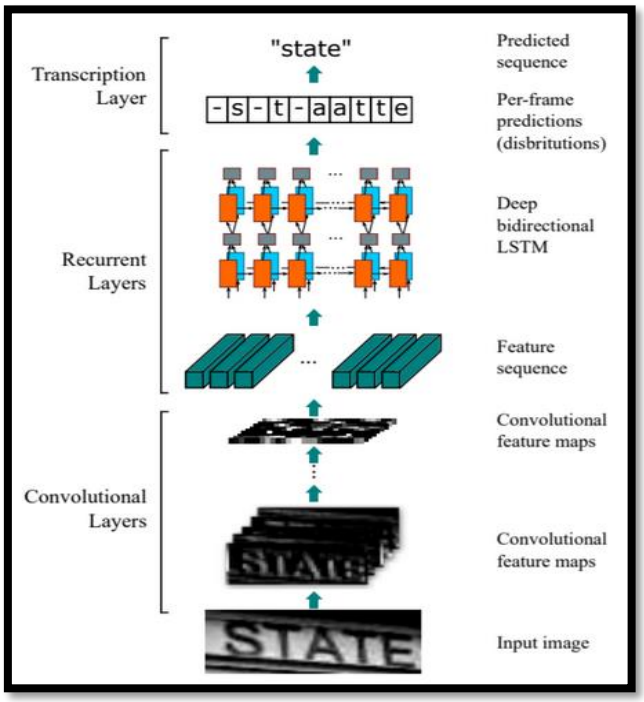
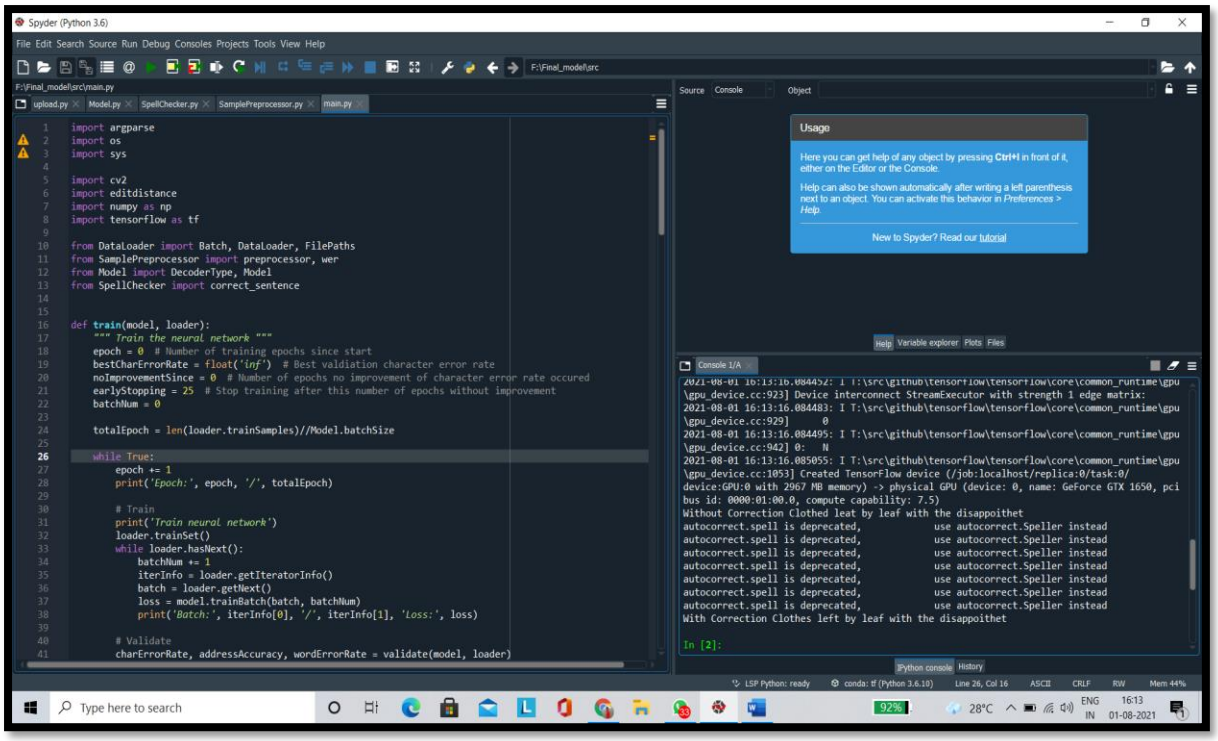
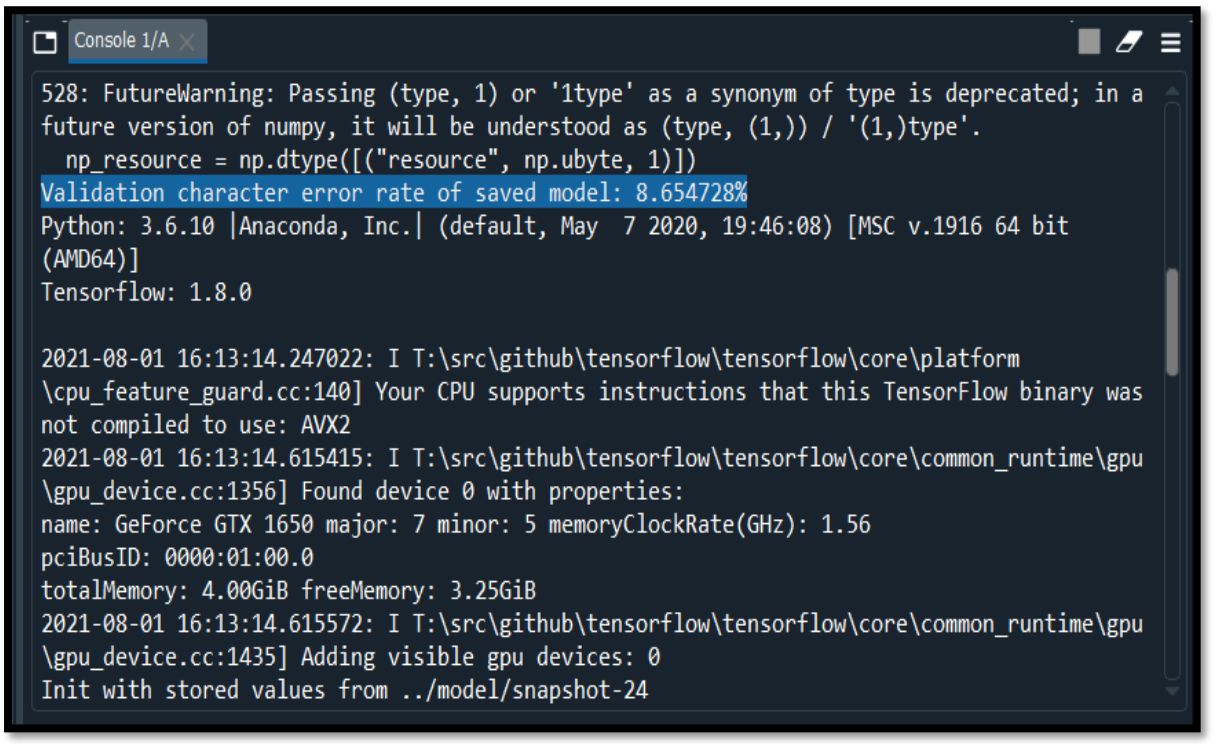


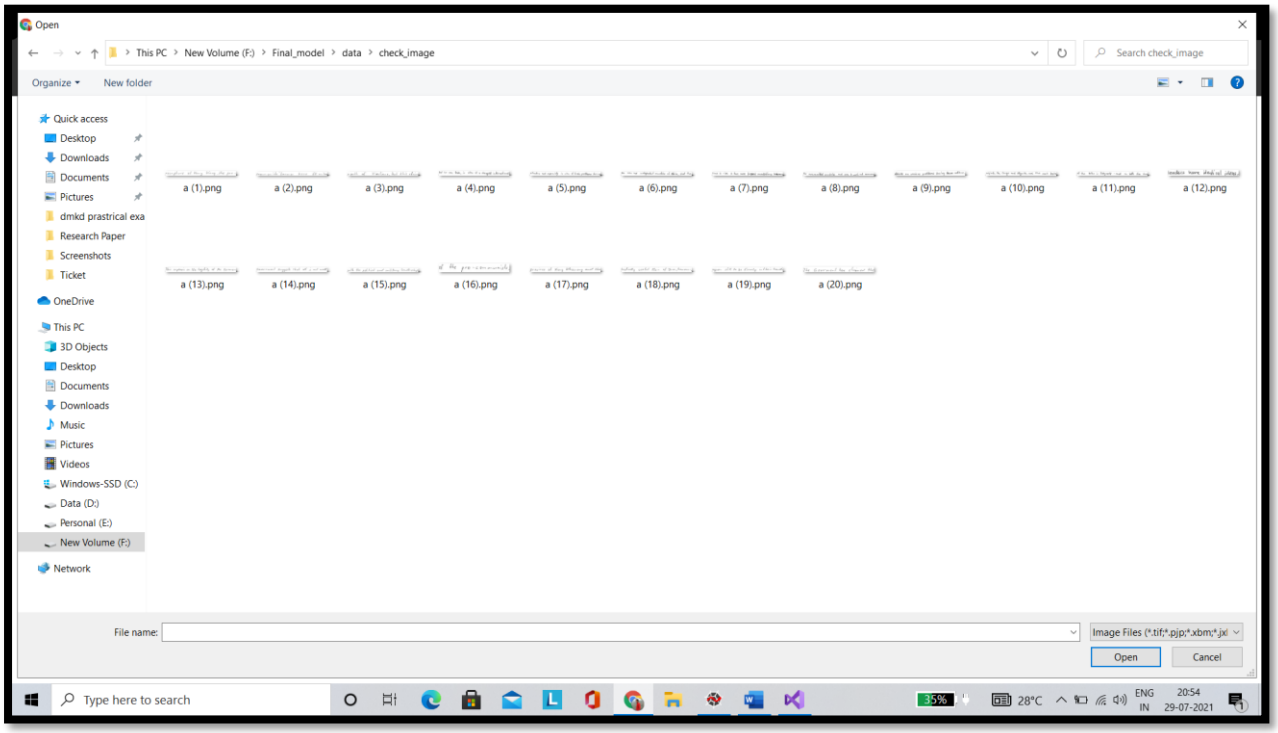
Fig 5: Architecture used in a Model

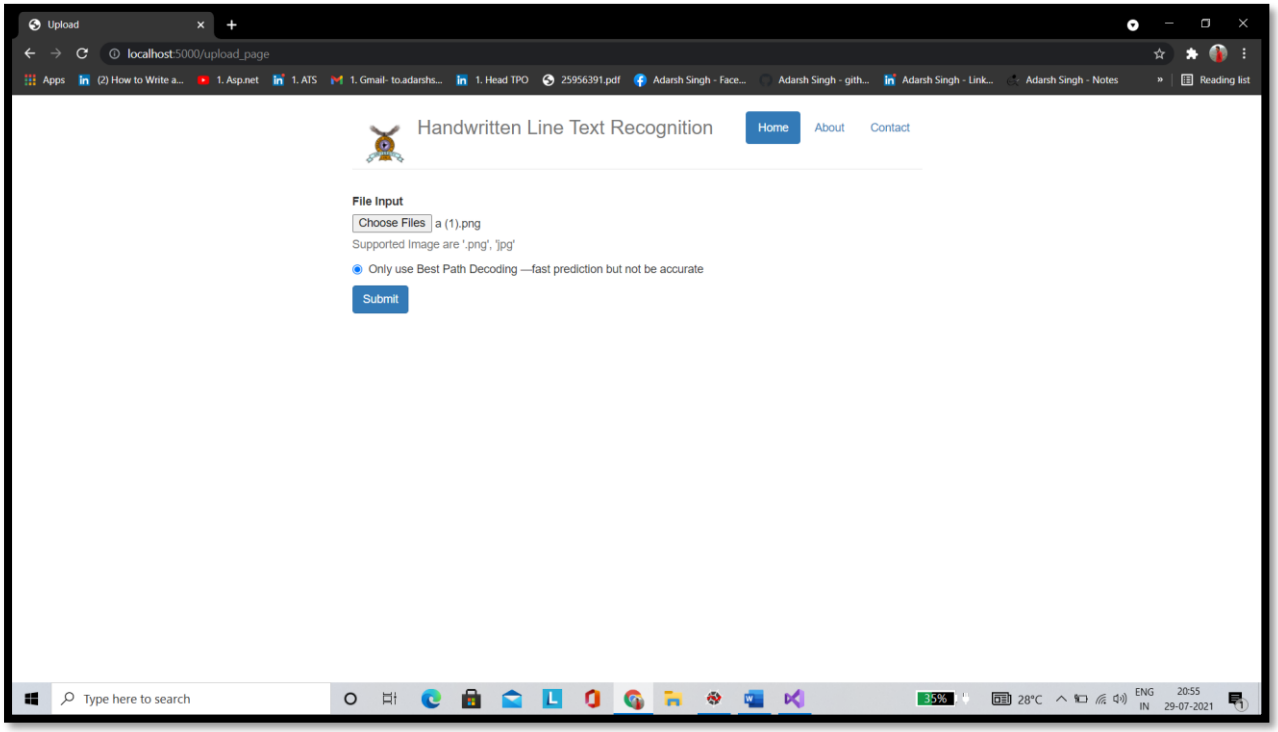
SYSTEMS TESTING:

Validation character error rate of saved model: 8.654728%


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
upload.py
Model.py
SpellChecker.py
SamplePreprocessor.py
main.py
import argparse
import os
import sys
import cv2
import editdistance
import numpy as np
import tensorflow as tf
from DataLoader import Batch, DataLoader, FilePaths
from SamplePreprocessor import preprocessor, wer
from Model import DecoderType, Model
from SpellChecker import correct_sentence
def train(model, loader):
""" Train the neural network """
epoch = 0 # Number of training epochs since start
bestCharErrorRate = float('inf') # Best validation character error rate
noImprovementSince = 0 # Number of epochs no improvement of character error rate occured
earlyStopping = 25 # Stop training after this number of epochs without improvement
batchNum = 0
totalEpoch = len(loader.trainSamples)//Model.batchSize
while True:
epoch += 1
print('Epoch:', epoch, '/', totalEpoch)
# Train
print('Train neural network')
loader.trainSet()
while loader.hasNext():
batchNum += 1
iterInfo = loader.getIteratorInfo()
batch = loader.getNext()
loss = model.trainBatch(batch, batchNum)
print('Batch:', iterInfo[0], '/', iterInfo[1], 'Loss:', loss)
# Validate
charErrorRate, addressAccuracy, wordErrorRate = validate(model, loader)
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Console 1/A
autocorrect.spell is deprecated, use autocorrect.Speller instead
With Correction Clothes left by leaf with the disappoithet
In [2]:
Type here to search

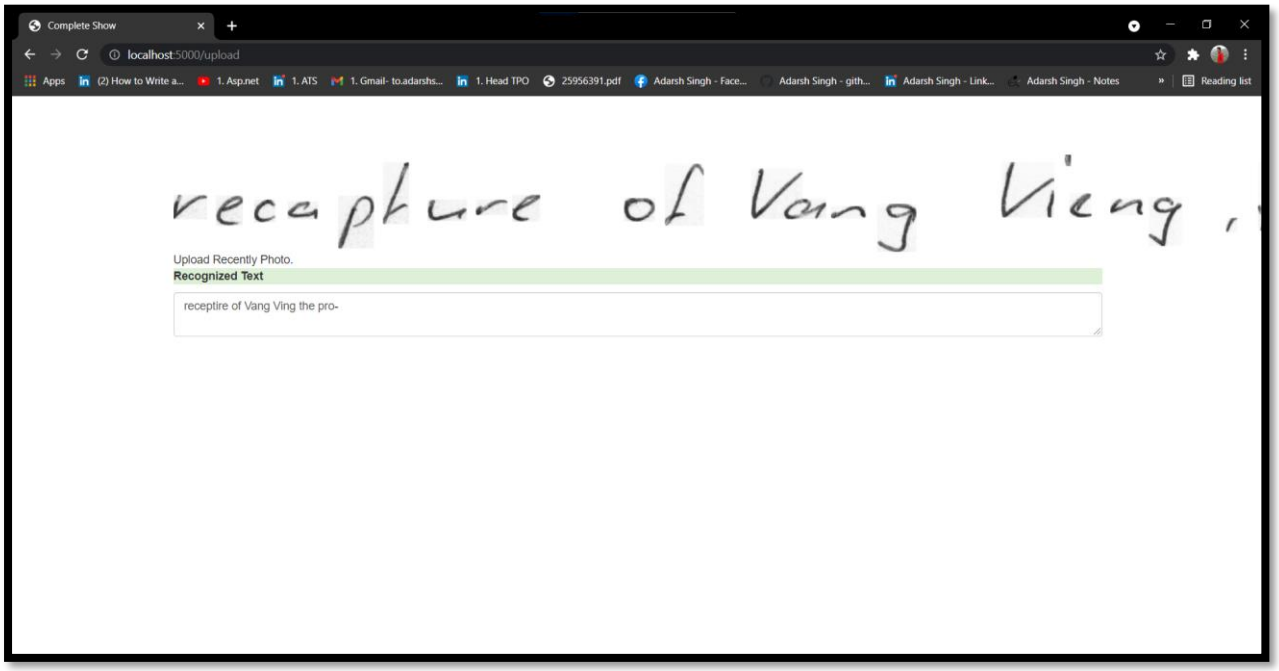

Console 1/A
528: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
np_resource = np.dtype([("resource", np.ubyte, 1)])
Validation character error rate of saved model: 8.654728%
Python: 3.6.10 |Anaconda, Inc.| (default, May 7 2020, 19:46:08) [MSC v.1916 64 bit (AMD64)]
Tensorflow: 1.8.0
2021-08-01 16:13:14.247022: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
2021-08-01 16:13:14.615415: I T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:1356] Found device 0 with properties:
name: GeForce GTX 1650 major: 7 minor: 5 memoryClockRate(GHz): 1.56
pciBusID: 0000:01:00.0
totalMemory: 4.00GiB freeMemory: 3.25GiB
2021-08-01 16:13:14.615572: I T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:1435] Adding visible gpu devices: 0
Init with stored values from ../model/snapshot-24


Open
This PC › New Volume (F:) › Final_model › data › check_image
Search check_image
Organize
New folder
Quick access
Desktop
Downloads
Documents
Pictures
dmkd prastrical exa
Research Paper
Screenshots
Ticket
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows-SSD (C:)
Data (D:)
Personal (E:)
New Volume (F:)
Network
a (1).png
a (2).png
a (3).png
a (4).png
a (5).png
a (6).png
a (7).png
a (8).png
a (9).png
a (10).png
a (11).png
a (12).png
a (13).png
a (14).png
a (15).png
a (16).png
a (17).png
a (18).png
a (19).png
a (20).png
File name:
Image Files (*.tif;*.pjp;*.xbm;*.jxl
Open
Cancel
Type here to search
35%
28°C
ENG
IN
20:54
29-07-2021

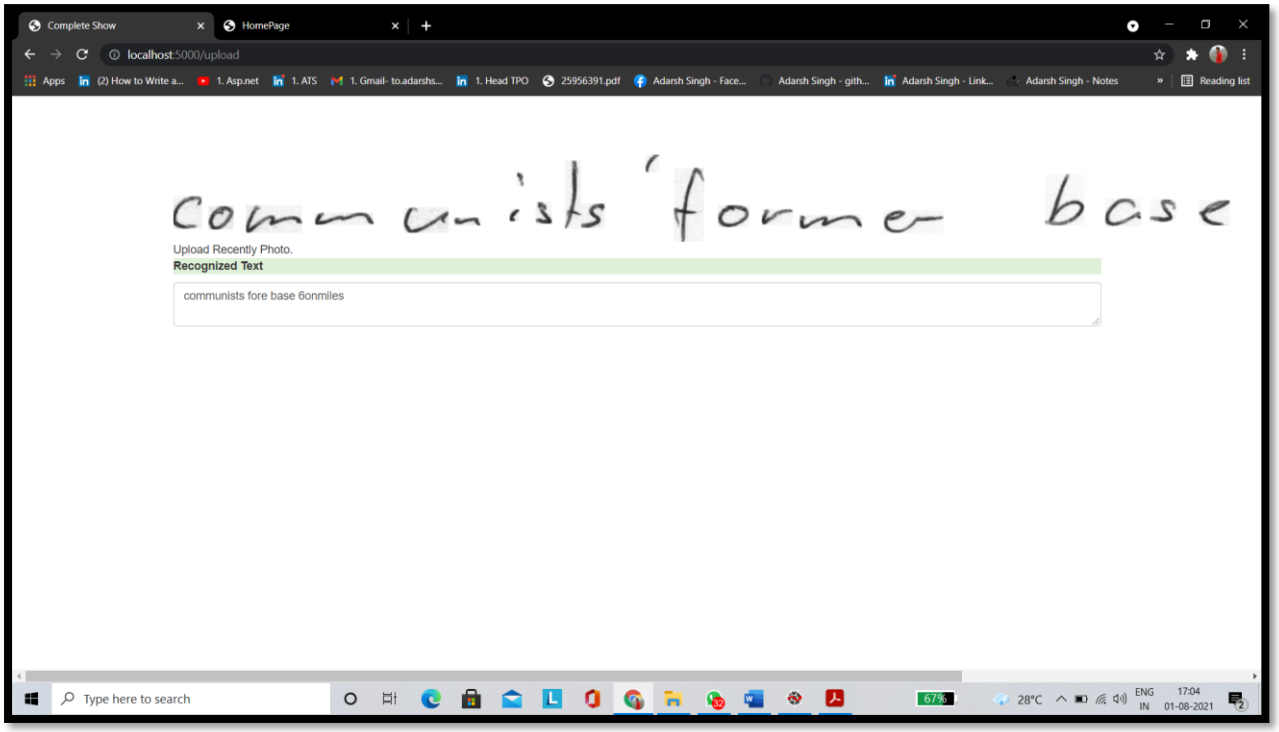

Upload
localhost:5000/upload_page
Apps
(2) How to Write a...
1. Asp.net
1. ATS
1. Gmail- to.adarshs...
1. Head TPO
25956391.pdf
Adarsh Singh - Face...
Adarsh Singh - gith...
Adarsh Singh - Link...
Adarsh Singh - Notes
Reading list
Handwritten Line Text Recognition
Home
About
Contact
File Input
Choose Files
a (1).png
Supported Image are '.png', 'jpg'
Only use Best Path Decoding —fast prediction but not be accurate
Submit
Type here to search
35%
28°C
ENG
IN
20:55
29-07-2021

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Line Image 1	Text Recognition	receptire of Vang Ving the pro-	receptire of Vang Ving the pro-
T02	Line Image 2	Text Recognition	communists fore base 60 miles	communists forward base 60 miles
T03	Line Image 4	Text Recognition	worth of Vientiane, bt this cief	north of Vientiane, but this claim
T04	Line Image 4	Text Recognition	Out he now feels, in view of a charyed international	But he now feels, in view of a changed international
T05	Line Image 5	Text Recognition	situation and especially in view of fresh problems facing	situation and especially in view of fresh problems facing
T06	Line Image 6	Text Recognition	he newaad independent countries of Africa, that the	The new and independent countries of Africa, that the
T07	Line Image 7	Text Recognition	time is riff to have more frequent cosullations between	time is ripe to have more frequent consultations between
T08	Line Image 8	Text Recognition	the encommitted countrile and every to work out common	the uncommitted countries and even to work out common
T09	Line Image 9	Text Recognition	fands an various problems fuciy those nations	stands a various problems facing those nations
T10	Line Image 10	Text Recognition	subjests, the congo and Aleria, are the main topios	subjects, the Congo and Aleria, are the main topics

Test Case: T01



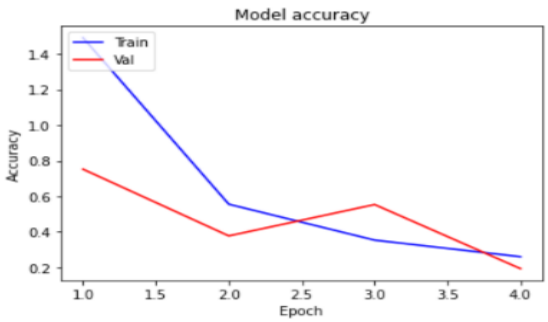
Test Case: T02



Loss vs Epochs Graph

It shows deacrease in loss over epochs.

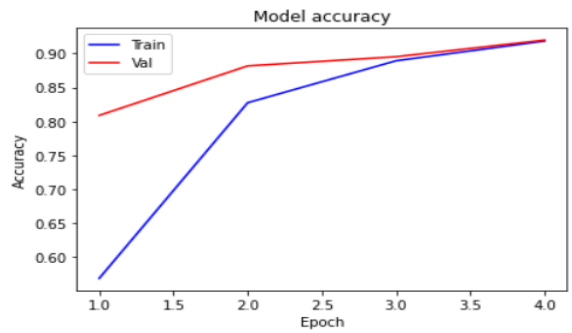
```
[ ]  plotgraph(epochs, loss, val_loss)
```



Accuracy vs Epochs Graph

It shows increase in accuracy over epochs.

```
[ ]  plotgraph(epochs, acc, val_acc)
```




```
In [ ]: #model.load_weights('gdrive/My Drive/check-04-0.2584.hdf5')
        scores = loaded_model.evaluate_generator(test_generator,280)
        print("Accuracy = ", scores[1], len(scores) )

        Accuracy =  0.9182981252670288 2

In [ ]: #model.load_weights('gdrive/My Drive/check-04-0.2584.hdf5')
        scores = model.evaluate_generator(test_generator,280)
        print("Accuracy = ", scores[1])

        Accuracy =  0.9174939393997192
```

CONCLUSION:

Utilizing cutting-edge techniques, for example, ANN, to perform and learn DL about by virtue

of which it handle the simple tasks that are completed in the fraction of a second, for example, content recognition, is only beginning to reveal the potential of AI. There are an infinite number of possible results and applications for this invention. Conventional recognition techniques used to function similarly to a biometric mechanism. Photograph sensor technology (Image Sensors) was used to collect matching points for real-world features and then convert them to data sets of recognised types. In any event, due to the availability of cutting-edge technologies like as ConvNets, we can filter and understand words with unprecedented accuracy.

We can get a high degree of accuracy when it comes to English content. With sufficient preparation and improvements, we could even align our model with the most stringent industry standards and preferences of Google and Amazon.

FUTURE SCOPE:

We could work on the model to make our model more diverse and expand its application to include other languages like Hindi, Marathi, and other regional languages we either need to find good quality datasets that are very well labelled and that could be used directly, or we need to devise a way to train our model without having labelled dataset.

We could work on improving the efficiency and accuracy by training the model more on the dataset and by using some more Deep Learning Algorithms.

REFERENCES:

[1]. Rohini G. Khalkar, Adarsh Singh Dikhit, Anirudh Goel, Manisha Gupta, "Handwritten Text Recognition using Deep Learning (CNN & RNN)", DOI: 10.17148/IARJSET.2021.86148, IARJSET, 2021, pp. 870-881

[2]. T. S. Gunawan, A. F. R. M. Noor, and M. Kartiwi, "Development of english handwritten recognition using deep neural network," 2018.

[3]. G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," axis preprint arXiv:1702.05373, 2017.

[4]. F. Cholletet al., "Keras," https://github.com/fchollet/keras, 2015.

[5]. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Man´e, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Vi´egas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015

[6]. C. Wu, W. Fan, Y. He, J. Sun, and S. Naoi, "Handwritten character recognition by alternately trained relaxation convolutional neural network," 2014 14th International Conference on Frontiers in Handwriting Recognition, pp. 291–296, 2014.

[7]. V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on. IEEE, 2014, pp. 285–290.

[8]. S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving offline handwritten text recognition with hybrid hmm/ann models," IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 4, pp. 767–779, 2011.

[9]. A. Graves and J. Schmidhuber, “Offline handwriting recognition with multidimensional recurrent neural networks,” in Advances in neural information processing systems, 2009, pp. 545–552.
[10]. L. Eikvil, “Optical character recognition,” citeseer. ist. psu. edu/142042. html, 1993.
[11] Kurhade, J. Naveenkumar, and A. K. Kadam, “An experimental on top-k high utility itemset mining by efficient algorithm Tkowithtku,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 8 Special Issue 3, pp. 519–522, 2019.
[12].S. D. Joshi, Dr. A. K. Kadam, Pritee Hulule, “A Survey Novel Approach for Efficient Selection of Test Case Prioritization Techniques,” vol. 3085, no. 12, pp. 999–1001, 2018.
[13].. A. K. Kadam, Amruta Magdum, prof. Dr. S. D Joshi, “A Survey on Test Case Prioritization with Rate of Fault Detection,” *Int. J. Res. Electron. Comput. Eng.*, vol. 6, no. 4, 2018.
[14]. N. Patil, A. Kadam, S. B. Wakurdekar, and S. D. Joshi, “Hybrid Approach of Code Analysis and Efforts Calculation for Software Reliability Growth Measurement and Cost Estimation,” *Iioab J.*, vol. 9, no. 2, SI, pp. 116–120, 2018.
[15]. Magdum, S. D. Joshi, A. K. Kadam, and A. Sarda, “Test case ranking with rate of fault finding,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 8 Special Issue 3, pp. 462–464, 2019.
[16]. K. Kadam, S. D. Joshi, D. Bhattacharyya, and H.-J. Kim, “Software Superiority Achievement through Functional Point and Test Point Analysis,” *Int. J. Softw. Eng. Its Appl.*, vol. 10, no. 11, pp. 181–192, 2016.
[17]. E. Pawar, A. K. Kadam, and S. D. Joshi, “Analysis of software reliability using testing time and testing coverage,” *Int. J. Adv. Res. Comp. Sci. Manag. Stud*, vol. 3, no. 5, pp. 143–148, 2015