HANDWRITTEN TEXT RECOGNITION

A Project work-I Report

Submitted in partial fulfillment of the requirement of the

Degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING

BY

Arushi Jain	**Avinash Kumar**	**Dhananjay Porwal**
EN19CS301076	**EN19CS301084**	**EN19CS301110**

Under the Guidance of

Prof. Snehal Moghe

Department of Computer Science & Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331

Aug – Dec 2022

Report Approval

The project work **"Handwritten Text Recognition"** is hereby approved as a creditable study of an engineering/computer application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the "Project Report" only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation

Affiliation

External Examiner

Name:

Designation

Affiliation

Declaration

We hereby declare that the project entitled **"Handwritten Text Recognition"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering completed under the supervision of **Mrs. Snehal Moghe, Assistant Professor Computer Science and Engineering,** Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, we declare that the content of this Project work, in full or in parts, has neither been taken from any other source nor has been submitted to any other Institute or University for the award of any degree or diploma.

Arushi Jain
24/11/2022

Avinash Kumar
24/11/2022

Dhananjay Porwal
24/11/2022

Certificate

I, **Snehal Moghe** certify that the project entitled **"Handwritten Text Recognition"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Arushi Jain, Avinash Kumar, and Dhananjay Porwal** is the record carried out by them under my guidance and that the work has not formed the basis of award of any other degree elsewhere.

_____

Snehal Moghe

Computer Science & Engineering

Medi-Caps University, Indore

_____

Dr. Pramod S. Nair

Head of the Department

Computer Science & Engineering

Medi-Caps University, Indore

Acknowledgments

I would like to express my deepest gratitude to the Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Dileep K. Patnaik,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm have always boosted up my morale. I also thank **Prof. (Dr.) D K Panda,** Pro Vice Chancellor, and **Dr. Suresh Jain,** Dean Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Pramod S. Nair** for his continuous encouragement for the betterment of the project.

I express my heartfelt gratitude to my Internal Project coordinator **Mrs. Snehal Moghe** without whose continuous help and support, this project would never have reached to the completion.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support, this report would not have been possible.

Arushi Jain, Avinash Kumar and Dhananjay Porwal
B.Tech. IV Year
Department of Computer Science & Engineering
Faculty of Engineering
Medi-Caps University, Indore

Executive Summary

Offline Handwritten Text Recognition (HTR) systems transcribe text contained in scanned images into digital text, an example is shown in Fig. 1. We will build a Neural Network (NN) which is trained on word-images from the IAM dataset. As the input layer (and therefore also all the other layers) can be kept small for word-images, NN-training is feasible on the CPU (of course, a GPU would be better). This implementation is the bare minimum that is needed for HTR using TF.

Get code and data

1. You need Python, TensorFlow, numpy and OpenCV installed
2. Further instructions (how to get the IAM dataset, command line parameters, …) can be found in the *README.md*
3. You can also install docker and run docker image to run virtual environment with all the required software with respective version used in this project.

NN which is able to recognize text in images. The NN consists of 5 CNN and 2 RNN layers and outputs a character-probability matrix. This matrix is either used for CTC loss calculation or for CTC decoding. An implementation using TF is provided and some important parts of the code were presented

Table of Contents

CHAPTER - 1
INTRODUCTION

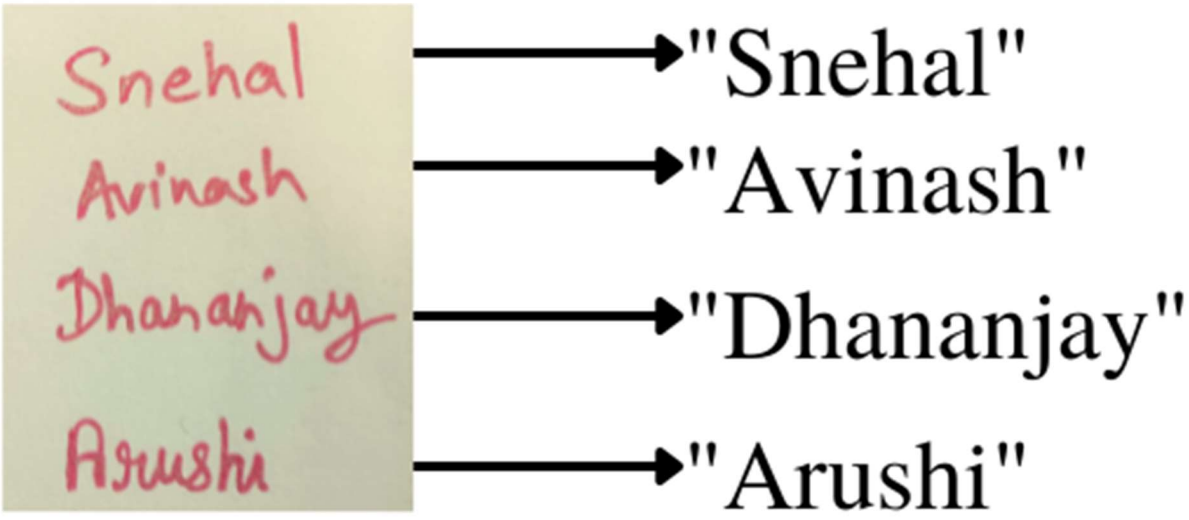
1.1 Introduction

Offline Handwritten Text Recognition (HTR) systems transcribe text contained in scanned images into digital text. We will build a Neural Network (NN) which is trained on word images from the IAM dataset.

The IAM Handwriting Database contains forms of handwritten English text which can be used to train and test handwritten text recognizers and to perform writer identification and verification experiments.

As the input layer can be kept small for word images, NN training is feasible on the CPU (of course, a GPU would be better). The word beam search decoder can be used instead of the two decoders shipped with TF. Words are constrained to those contained in a dictionary, but arbitrary non-word character strings (numbers, punctuation marks) can still be recognized.

The following illustration shows a sample of our model input and Output:



1.2 Literature Review

1.2.1 Understanding of a convolutional neural network | IEEE Conference Publication | IEEE Xplore

One of the most popular deep neural networks is the Convolutional Neural Network (CNN). CNN has an excellent performance in machine learning problems. Specially the applications that deal with image data, such as the largest image classification data set (Image Net), computer vision, and natural language processing (NLP) and the results achieved were very amazing.

1.2.2 IET Digital Library: Handwriting recognition using CNN and its optimization approach (theiet.org) Publication Date: October 2021

1.2.3 Exploration of CNN Features for Online Handwriting Recognition | IEEE Conference Publication | IEEE Xplore Published in 2019 International Conference on Document Analysis and Recognition (ICDAR)

1.2.4 Offline Handwritten Numeral Recognition using Combination of Different Feature Extraction Techniques

A handwritten numeral recognition system using a combination of different feature extraction techniques has been presented in this paper. SVM classifier has been considered for classification purposes. For experimental results, 6000 samples of isolated handwritten numerals have been considered. The proposed system achieves maximum recognition accuracy of 96.3% using a five-fold cross-validation technique.

1.2.5 Handwritten Character Recognition in English: A Survey

This paper presents a comprehensive review of Handwritten Character Recognition (HCR) in the English language. Handwritten character recognition has been applied in a variety of applications like Banking sectors, Health care industries, and many such organizations where handwritten documents are dealt with. Handwritten Character Recognition is the process of conversion of handwritten text into machine-readable form. For handwritten characters, there are difficulties it differs from one writer to another, even when the same person writes the same character there is a difference in shape, size, and position of the character. The latest

research in this area has used different types of methods, classifiers, and features to reduce the complexity of recognizing handwritten text.

1.3 Problem Definition

People make notes on pen and paper. Everyone does not have the access to products through which they can create handwritten notes in a digital file. They have to manually type the written notes into a digital file. It is a very cumbersome process as it wastes manpower and time.

1.4 Objective

To provide an easy-to-use application that would convert handwritten images into digital and editable form. To save time and resources by providing a platform that will convert a tedious manual task into an easy-to-click process. We discussed a NN which is able to recognize text in images. The NN consists of 5 CNN and 2 RNN layers and outputs a character-probability matrix. This matrix is either used for CTC loss calculation or for CTC decoding. An implementation using TF is provided and some important parts of the code were presented. Finally, hints to improve the recognition accuracy were given.

1.5 Methodology

1.5.1 Deep Learning

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign or to distinguish a pedestrian from a lamppost. Deep learning is a branch of machine learning that uses neural networks with many layers. A deep neural network analyzes data with learned representations similar to the way a person would look at a problem.

1.5.2 Convolutional Neural Network (CNN)

It is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. It is specifically used for image recognition and tasks that involve the processing of pixel data.

1.5.3 Recurrent Neural Network(RNN)

Recurrent Neural Network is a type of neural network used to deal specifically with sequential data. RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

1.5.4 Git and Github

Git is a version control system used for tracking changes in computer files. It is generally used for source code management in software development. Git is used to track changes in the source code. Usually used for coordinating work among programmers collaboratively developing source code during software development.

1.5.5 Streamlit*

Streamlit is an open-source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as Scikitlearn , Keras , PyTorch , SymPy(latex) , NumPy , pandas , Matplotlib , etc.

1.5.6 Docker*

It is an open platform for developing, shipping, and running applications. Docker enables us to separate our applications from our infrastructure so, we can deliver software quickly. With Docker, we can manage our infrastructure in the same ways we manage our applications.

Docker: Experimental Feature | Streamlit: Experimental Feature

1.6 Source of Data

- Kaggle.
- IAM Datasets.
- Scrapped from different websites.
- Created by own.

CHAPTER – 2

Report on Present Investigation

2.1 Experimental Setup

To build this project we used several technologies and frameworks:

Hardware Dependencies:

Hardware	Capacity
ROM	Up to 20 GB
RAM	8 GB DDR4
CPU	Intel Core i5 12th Gen
GPU	Nvidia GeForce GTX 1650

Software Dependencies:

OS: Windows 11/10 & Ubuntu 22.04 LTS

Software	Version
Python	3.11.0
Docker	20.10.21
editDistance	0.5.2
lmdb	1.0.0
matplotlib	3.2.1
numpy	1.19.5

Opencv	4.4.0.46
Path	15.0.0
tensorflow	2.4.0

2.2 Procedure Adopted

To complete our project, we have followed an iterative model. Iterative model gives an exact performance of the development of software as a life cycle. It primarily focuses on preliminary growth and design and then gains momentum slowly with more complexity as well as meet requirements until the final software is built entirely. So, basically, the iterative development model is an approach of segmenting any large software development process into smaller portions. There are major reasons to use this SDLC like:

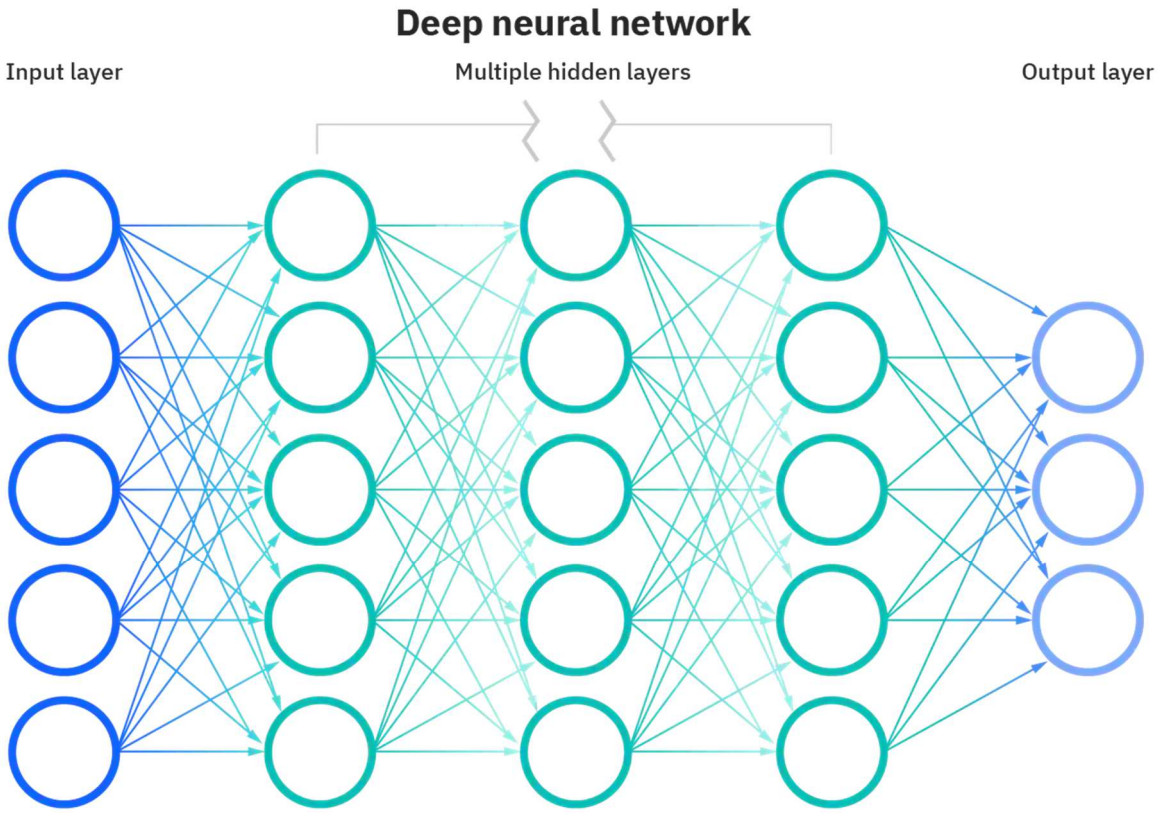
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.

CHAPTER – 3

Tools and Technology

3.1 Neural Network

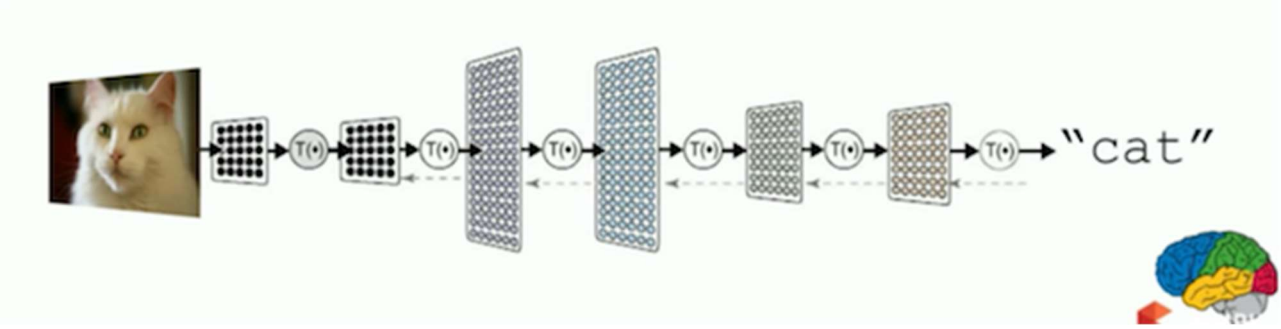
A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. It creates an adaptive system that computers use to learn from their mistakes and improve continuously. Thus, artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy.



Neural networks can help computers make intelligent decisions with limited human assistance. This is because they can learn and model the relationships between input and output data that are nonlinear and complex.

Neural networks are being applied to many real-life problems today, including speech and image recognition, spam email filtering, finance, and medical diagnosis, to name a few.

Machine learning algorithms that use neural networks generally do not need to be programmed with specific rules that define what to expect from the input. The neural net learning algorithm instead learns from processing many labeled examples (i.e. data with with "answers") that are supplied during training and using this answer key to learn what characteristics of the input are needed to construct the correct output. Once a sufficient number of examples have been processed, the neural network can begin to process new, unseen inputs and successfully return accurate results. The more examples and variety of inputs the program sees, the more accurate the results typically become because the program learns with experience.



Neural networks can be applied to a broad range of problems and can assess many different types of input, including images, videos, files, databases, and more. They also do not require explicit programming to interpret the content of those inputs.

Because of the generalized approach to problem solving that neural networks offer, there is virtually no limit to the areas that this technique can be applied.

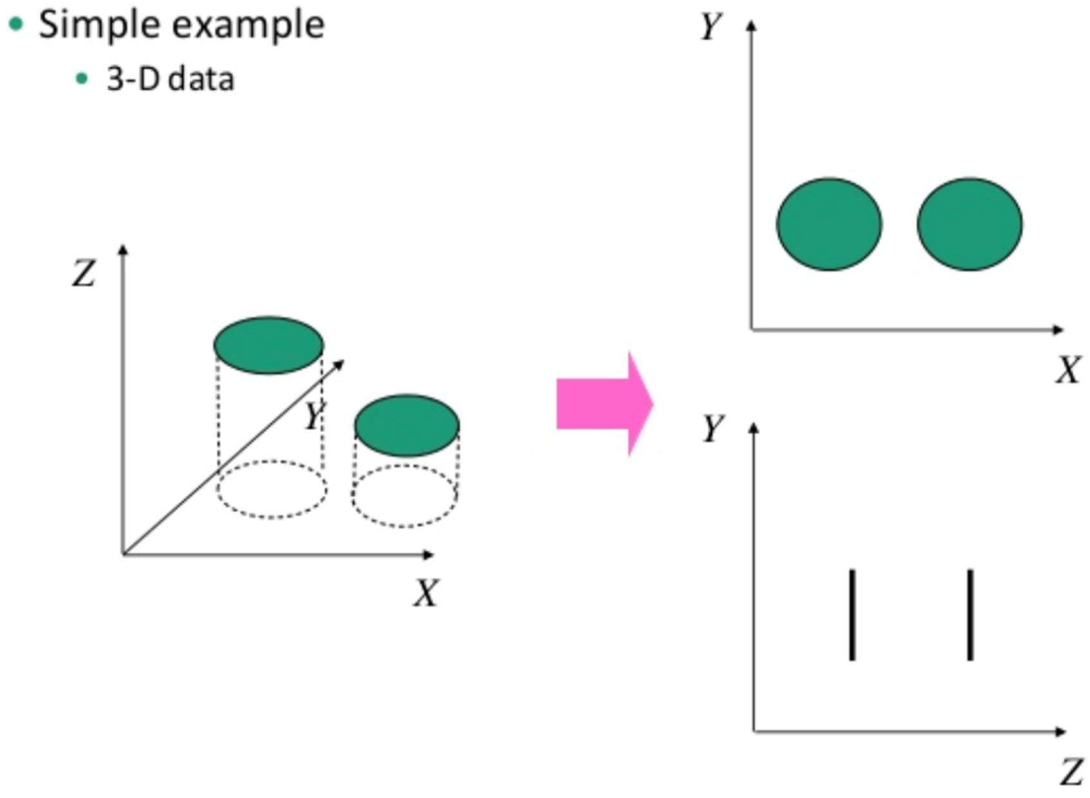
3.2 Dimensionality Reduction

The number of input features, variables, or columns present in a given dataset is known as dimensionality.

Dimensionality reduction simply refers to the process of reducing the number of attributes in a dataset while keeping as much of the variation in the original dataset as possible. It is a data preprocessing step meaning that we perform dimensionality reduction before training the model.

When we reduce the dimensionality of a dataset, we lose some percentage (usually 1%-15% depending on the number of components or features that we keep) of the variability in the original data. But, don't worry about losing that much percentage of the variability in the original data because dimensionality reduction will lead to the following advantages:

- A lower number of dimensions in data means less training time and less computational resources and increases the overall performance of machine learning algorithms.
- Dimensionality reduction avoids the problem of overfitting.
- Dimensionality reduction is extremely useful for data visualization.
- Dimensionality reduction removes noise in the data.
- Dimensionality reduction can be used for image compression.
- Dimensionality reduction can be used to transform non-linear data into a linearly-separable form.



The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

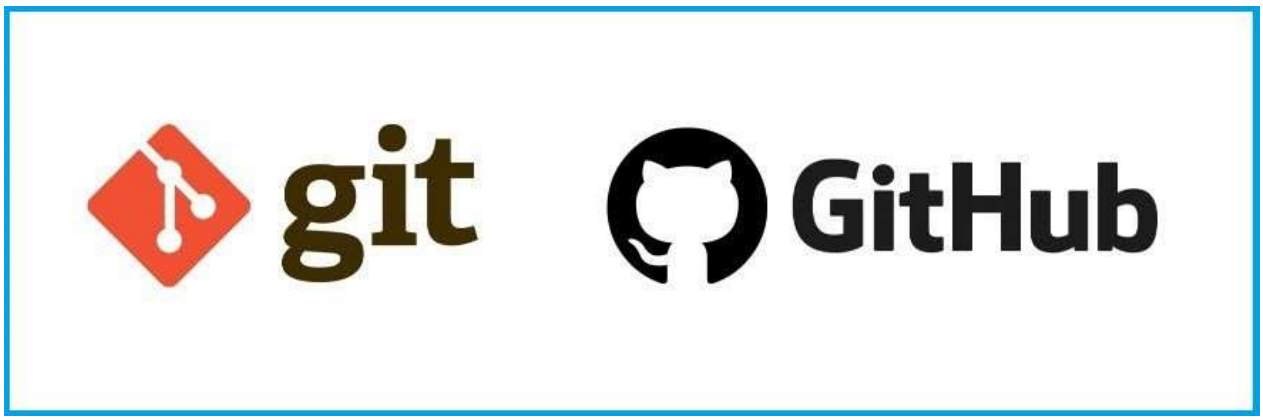
3.3 Git and GitHub

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is a version control system used for tracking changes in computer files. It is generally used for source code management in software development.

- Git is used to track changes in the source code
- The distributed version control tool is used for source code management
- It allows multiple developers to work together
- It supports non-linear development through its thousands of parallel branches.

Git is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.



GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.

GitHub hosts Git repositories and provides developers with tools to ship better code through command line features, issues (threaded discussions), pull requests, code reviews, or the use of a collection of free and for-purchase apps in the GitHub Marketplace. With collaboration layers like the GitHub flow, a community of 15 million developers, and an ecosystem with hundreds of integrations, GitHub changes the way software is built.

git and GitHub are not the same things. Git is an open-source, version control tool created in 2005 by developers working on the Linux operating system; GitHub is a company founded in

2008 that makes tools that integrate with git. You do not need GitHub to use git, but you cannot use GitHub without using git.

3.4 Docker*

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

The benefits of Docker in building and deploying applications are many:

- Caching a cluster of containers
- Flexible resource sharing
- Scalability - many containers can be placed in a single host
- Running your service on hardware that is much cheaper than standard servers
- Fast deployment, ease of creating new instances, and faster migrations.
- Ease of moving and maintaining your applications
- Better security, less access needed to work with the code running inside containers, and fewer software dependencies

3.5 Streamlit*

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy powerful data apps.

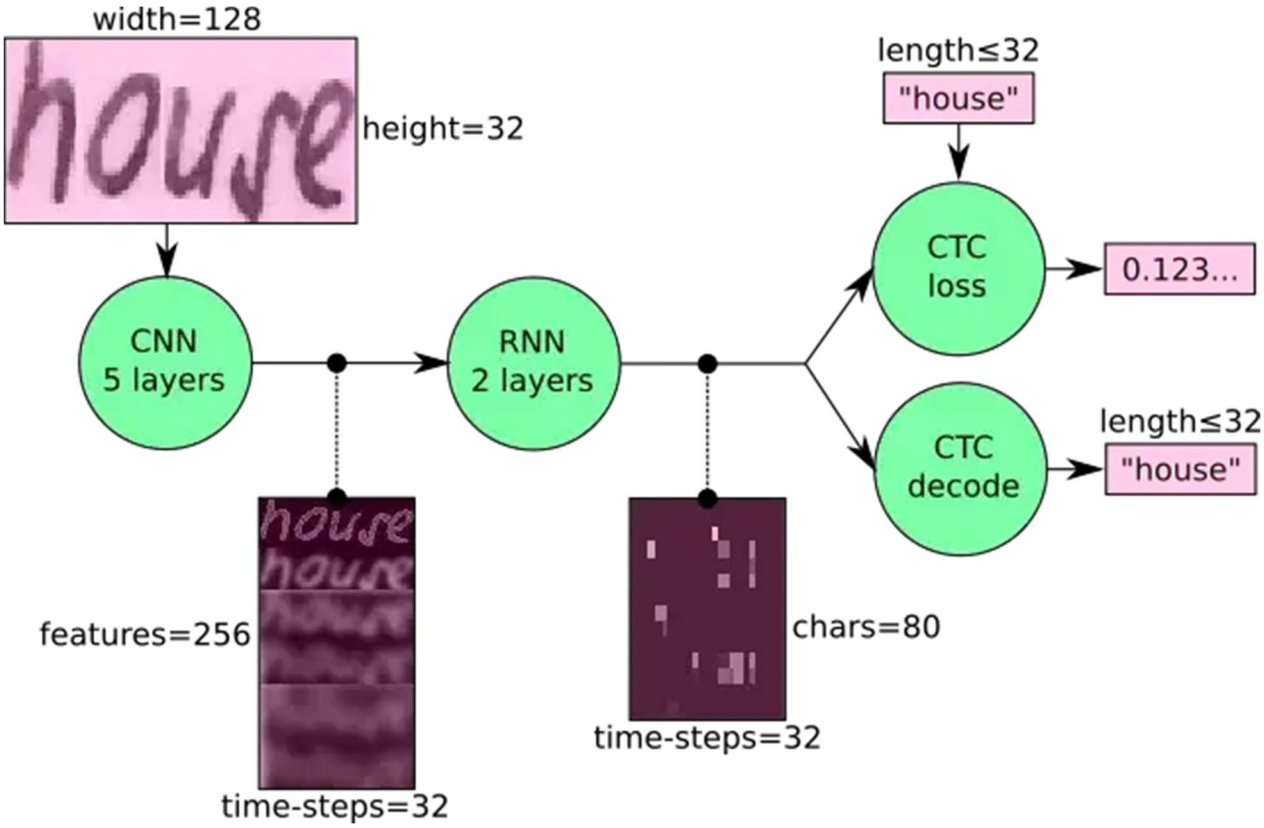
It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as Scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib, etc.

Streamlit turns data scripts into shareable web apps in minutes. All in pure Python. No front-end experience required. The core benefit of a tool like Streamlit is that it is Python based. Many of the insight graphs, evaluation graphs, and prediction graphs have already been created by data scientists during model development. This usually happens in the form of Jupyter notebooks. Moving the code to generate these visualizations into a Streamlit dashboard requires little work. Streamlit allows you to re-use any Python code you have already written. This can save considerable amounts of time compared to non-Python based tools where all code to create visualizations needs to be re-written.

CHAPTER 4
MODEL OVERVIEW

4.1 Overview

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.



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

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

Equation.1

4.2 Operations

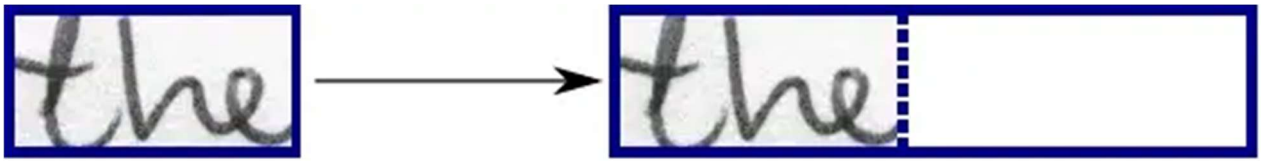
CNN: The input image is fed into the CNN layers. These layers are trained to extract relevant features from the image. Each layer consists of three operations. First, the convolution operation, applies a filter kernel of size 5×5 in the first two layers and 3×3 in the last three layers to the input.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

RNN: The feature sequence contains 256 features per time-step, the RNN propagates relevant information through this sequence. The popular Long Short-Term Memory (LSTM) implementation of RNNs is used, as it is able to propagate information through longer distances and provides more robust training characteristics than vanilla RNNs. The RNN output sequence is mapped to a matrix of size 32×80. The IAM dataset consists of 79 different characters, further one additional character is needed for the CTC operation (CTC blank label), therefore there are 80 entries for each of the 32-time steps.

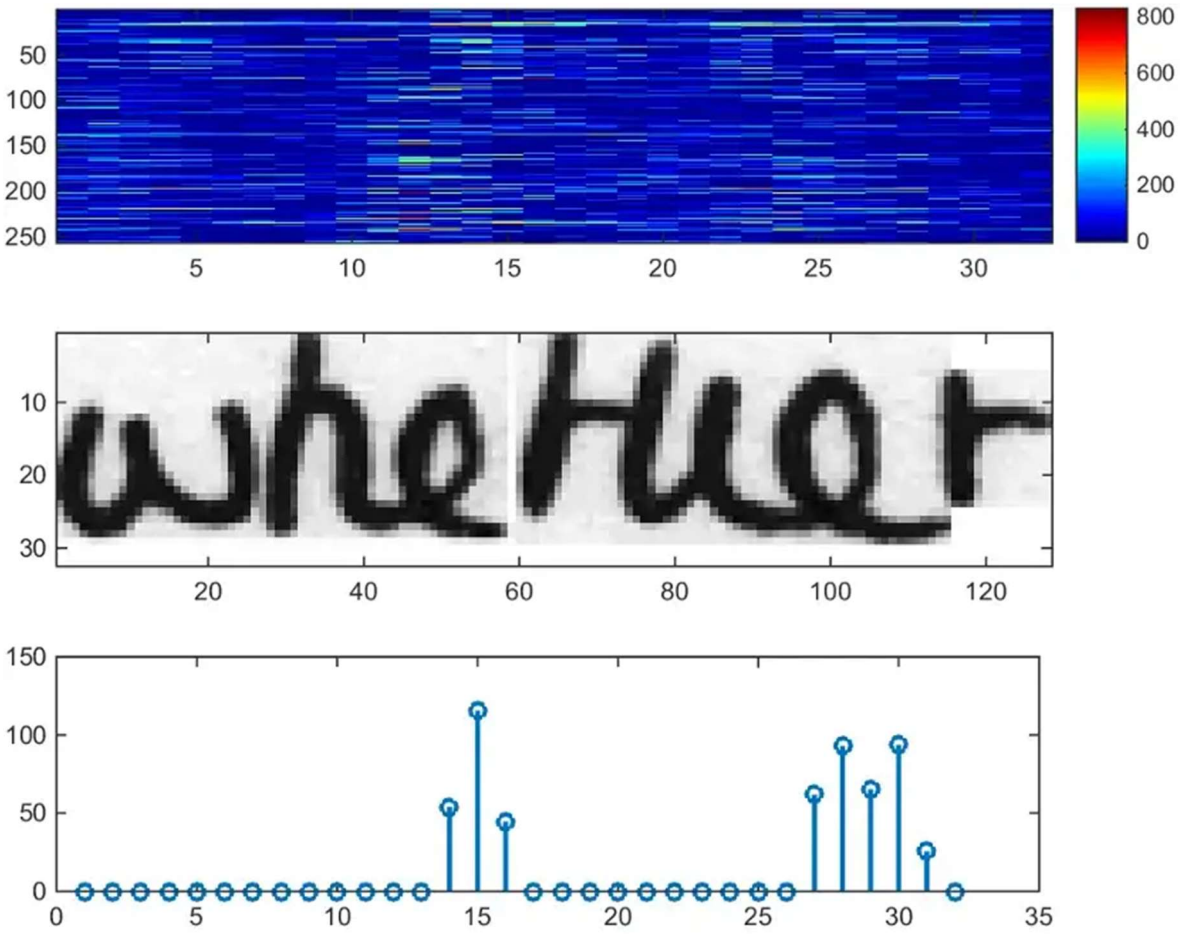
CTC: While training the NN, the CTC is given the RNN output matrix and the ground truth text and it computes the loss value. While inferring, the CTC is only given the matrix and it decodes it into the final text. Both the ground truth text and the recognized text can be at most 32 characters long.

4.3 Data

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

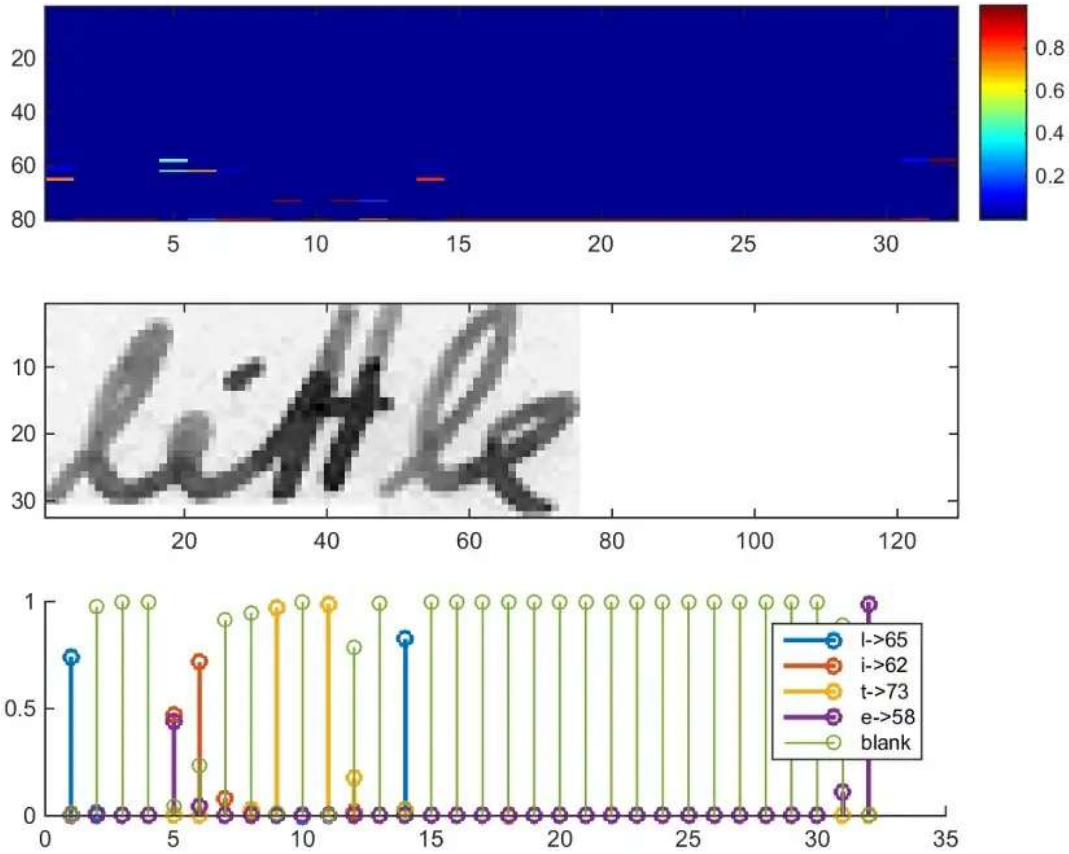


CNN output:



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

RNN output:



This figure shows a visualization of the RNN output matrix for an image containing the text "little". The matrix shown in the top-most graph contains the scores for the characters including the CTC blank label as its last (80th) entry. The other matrix-entries, from top to bottom, correspond to the following characters: "!"#&'()*+,-./0123456789:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".

It can be seen that most of the time, the characters are predicted exactly at the position they appear in the image (e.g. compare the position of the "i" in the image and in the graph). Only the last character "e" is not aligned. But this is OK, as the CTC operation is segmentation-free

and does not care about absolute positions. From the bottom-most graph showing the scores for the characters “l”, “i”, “t”, “e” and the CTC blank label, the text can easily be decoded: we just take the most probable character from each time step, this forms the so-called best path, then we throw away repeated characters and finally all blanks: “l---ii--t-t--l-…-e”.

4.4 Implementation using TF

The implementation consists of 4 modules:

1. SamplePreprocessor.py: prepares the images from the IAM dataset for the NN
2. DataLoader.py: reads samples, put them into batches, and provides an iterator interface to go through the data
3. Model.py: creates the model as described above, loads and saves models, manages the TF sessions, and provides an interface for training and inference
4. main.py: puts all previously mentioned modules together

We only look at Model.py, as the other source files are concerned with basic file IO (DataLoader.py) and image processing (SamplePreprocessor.py).

CNN

For each CNN layer, create a kernel of size k×k to be used in the convolution operation.

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

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

RNN

Create and stack two RNN layers with 256 units each.

Then, create a bidirectional RNN from it, such that the input sequence is traversed from front to back and the other way round. As a result, we get two output sequences fw and bw of size 32×256, which we later concatenate along the feature-axis to form a sequence of size 32×512. Finally, it is mapped to the output sequence (or matrix) of size 32×80 which is fed into the CTC layer.

CTC

For loss calculation, we feed both the ground truth text and the matrix to the operation. The ground truth text is encoded as a sparse tensor. The length of the input sequences must be passed to both CTC operations.

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

4.5 Training

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

4.6 Improving the model (Not Implemented Yet)

In case you want to feed complete text lines as shown in Fig. 6 instead of word images, you have to increase the input size of the NN.

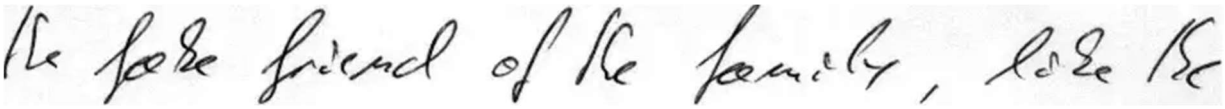


Fig: A complete text line can be fed into the NN if its input size is increased (image taken from IAM).

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

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

Remove the cursive writing style in the input images.

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

Add more CNN layers

Replace LSTM with 2D-LSTM

Decoder: use token passing or word beam search decoding to constrain the output to dictionary words.
Text correction: if the recognized word is not contained in a dictionary, search for the most similar one.

CHAPTER 5

Challenges in Handwriting Recognition

- Huge variability and ambiguity of strokes from person to person.
- The handwriting style of an individual person also varies from time to time and is inconsistent.
- Poor quality of the source document/image due to degradation over time.
- Text in printed documents sit in a straight line whereas humans need not write a line of text in a straight line on white paper.
- Cursive handwriting makes separation and recognition of characters challenging
- Text in handwriting can have variable rotation to the right which is in contrast to the printed text where all the text sits up straight.
- Collecting a well labelled dataset to learn is not cheap compared to synthetic data.

CHAPTER 6

Result and Discussions

We discussed a NN which is able to recognize text in images. The NN consists of 5 CNN and 2 RNN layers and outputs a character-probability matrix. This matrix is either used for CTC loss calculation or for CTC decoding.

Although there have been significant developments in technology that help in better recognition of handwritten textbooks, HTR is far from a answered problem compared to OCR and hence isn't yet considerably employed in assiduity. nonetheless with the pace of technology elaboration and with the preface of models like mills, we can anticipate HTR models to come a commonplace soon. We discussed a NN which is able to recognize text in images. The NN consists of 5 CNN and 2 RNN layers and outputs a character-probability matrix. This matrix is either used for CTC loss calculation or for CTC decoding. An implementation using TF is provided and some important parts of the code were presented. Finally, hints to improve the recognition accuracy were given. The use of binarization features along with the neural network classifier employing back-propagation algorithm delivers outstanding classification accuracy of 77.69 %. Training sample quality, feature extraction technique and the classifier are the main factors deciding the accuracy of the recognition system. All these techniques can be refined because a scope of improvement is always there. In future, a combination of binarization features with some other type of features such as Projection profile Features, can be investigated in the recognition experiment. Apart from MLP classifier, other classifiers such as RBF, HMM, SVM etc. can also be examined in future.

CHAPTER 7

Use Case and Future Scope

7.1 Healthcare and pharmaceuticals

Patient prescription digitization is a major pain point in healthcare/pharmaceutical industry. For example, Roche is handling millions of petabytes of medical PDFs daily. Another area where handwritten text detection has key impact is patient enrollment and form digitization. By adding handwriting recognition to their toolkit of services, hospitals/pharmaceuticals can significantly improve user experience

7.2 Insurance

A large insurance industry receives more than 20 million documents a day and a delay in processing the claim can impact the company terribly. The claims document can contain various different handwriting styles and pure manual automation of processing claims is going to completely slow down the pipeline

7.3 Banking

People write cheques on a regular basis and cheques still play a major role in most non-cash transactions. In many developing countries, the present cheque processing procedure requires a bank employee to read and manually enter the information present on a cheque and also verify the entries like signature and date. As a large number of cheques have to be processed every day in a bank a handwriting text recognition system can save costs and hours of human work

7.4 Online Libraries

Huge amounts of historical knowledge is being digitized by uploading the image scans for access to the entire world. But this effort is not very useful until the text in the images can be identified which can be indexed, queried and browsed. Handwriting recognition plays a key role in bringing alive the medieval and 20th century documents, postcards, research studies etc.

Bibliography

- Build a Handwritten Text Recognition System using TensorFlow | by Harald Scheidl | Towards Data Science
- Handwritten Text Recognition in Historical Documents (tuwien.at)
- 1835 (tuwien.at)
- GeeksforGeeks | A computer science portal for geeks
- 3.11.0 Documentation (python.org)
- API Documentation | TensorFlow v2.11.0
- Streamlit documentation
- What a text recognition system actually sees | by Harald Scheidl | Towards Data Science
- An Intuitive Explanation of Connectionist Temporal Classification | by Harald Scheidl | Towards Data Science
- Beam Search Decoding in CTC-trained Neural Networks | by Harald Scheidl | Towards Data Science
- Word Beam Search: A CTC Decoding Algorithm | by Harald Scheidl | Towards Data Science
- Handwritten Text Recognition in Historical Documents (tuwien.at)
- [1507.05717] An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition (arxiv.org)
- Theodore Bluche - Research Engineer (tbluche.com)
- Index - Obsidian Help
- Docker Documentation | Docker Documentation
- Git - Documentation (git-scm.com)
- FAQ: Build a Handwritten Text Recognition System using TensorFlow | by Harald Scheidl | Towards Data Science