

A
Internship Report on
**DEEP LEARNING AIDED HAND WRITTEN CHARACTER
RECOGNITION**

Submitted in partial fulfillment of the requirements for the award of
Degree of

BACHELOR OF TECHNOLOGY

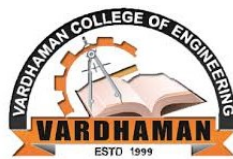
In

INFORMATION TECHNOLOGY

By

MYLAVARAPU SOUMYA (18881A1242)

Under the Guidance of
Dr.Muni SekharVelpuru
Associate Professor &
Head of the Department
Department of Information Technology



**DEPARTMENT OF INFORMATION TECHNOLOGY
VARDHAMAN COLLEGE OF ENGINEERING
(AUTONOMOUS)**

(Affiliated to JNTUH, Approved by AICTE and Accredited by NBA)
Shamshabad - 501 218, Hyderabad

DECLARATION

I hereby declare that the work described in this thesis entitled “**Deep Learning Aided Hand Written Character Recognition**” which is being submitted by me in partial fulfillment for the award of **BACHELOR OF TECHNOLOGY** in the Department of Information Technology, Vardhaman College of Engineering to the Jawaharlal Nehru Technological University Hyderabad.

The work is original and has not been submitted for any Degree or Diploma of this or any other university.

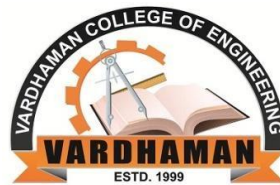
Signature of the Student

M Soumya
(18881A1242)

**DEPARTMENT OF INFORMATION TECHNOLOGY
VARDHAMAN COLLEGE OF ENGINEERING
(AUTONOMOUS)**

(Affiliated to JNTUH, Approved by AICTE and Accredited by NBA)
Shamshabad - 501 218, Hyderabad

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the Internship report entitled, “**Deep Learning Aided Hand Written Character Recognition**”, done by **Mylavarapu Soumya(18881A1242)**, Submitted to the Department of Information Technology, **VARDHAMAN COLLEGE OF ENGINEERING**, in partial fulfillment of the requirements for the Degree of **BACHELOR OF TECHNOLOGY** in **Information Technology**, during the year 2020-21. It is certified that she has completed the project satisfactorily.

Signature of Supervisor

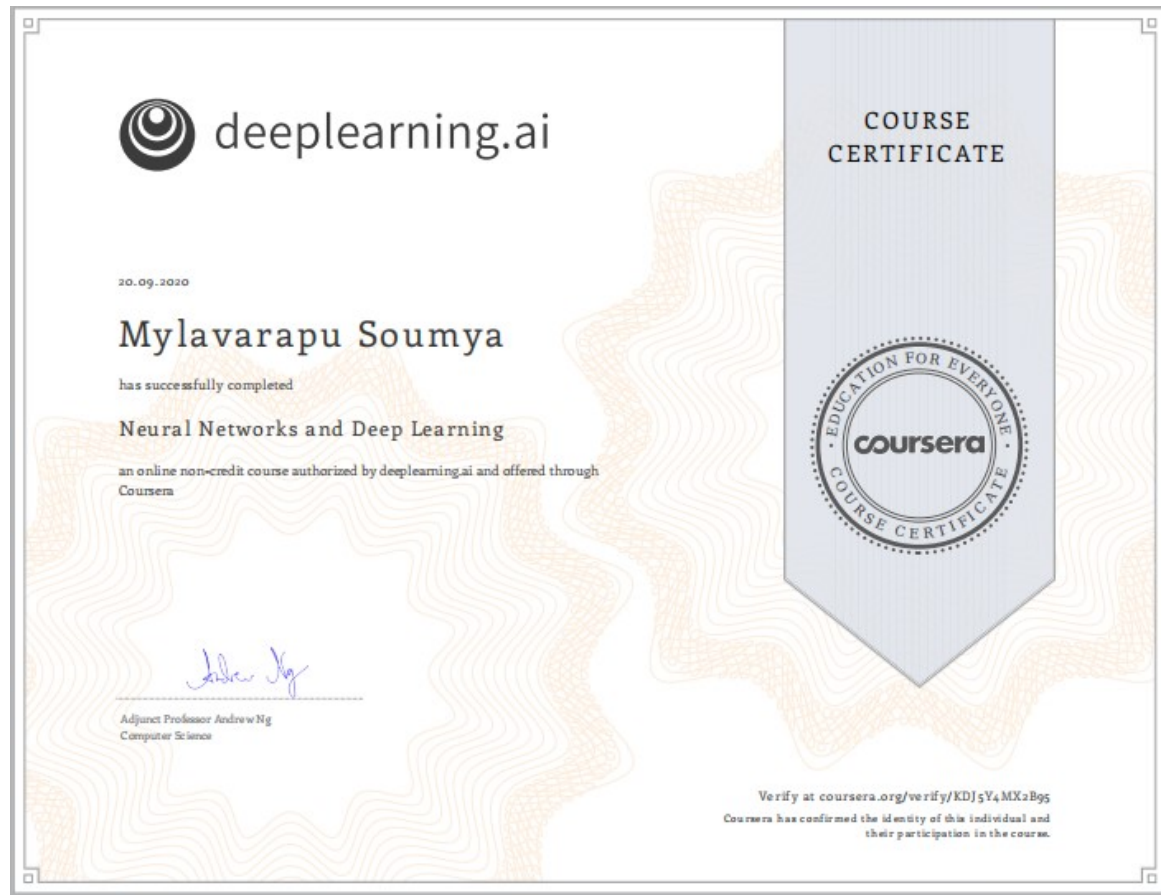
Dr. Muni SekharVelpuru
Associate Professor & Head

Signature of Head of the Department

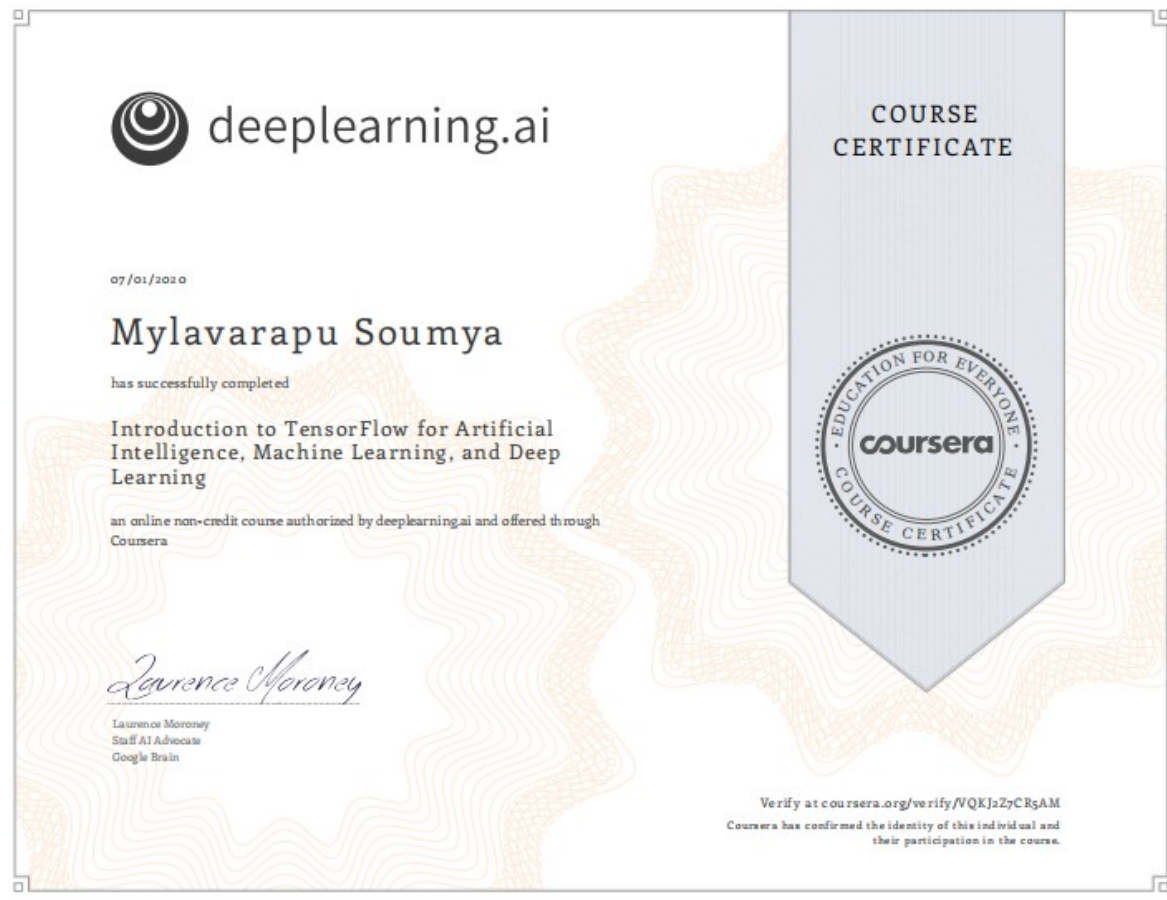
Dr. Muni SekharVelpuru
Associate Professor & Head

CERTIFICATION

Neural Networks and Deep Learning



Introduction to Tensor Flow for Artificial Intelligence, Machine Learning, Deep Learning



ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

I express my heartfelt thanks to **Dr. Muni SekharVelpuru**, Associate Professor, Head & Internship Project Supervisor, for his suggestions in the selection and carrying out an in-depth study of the topic. His valuable guidance and encouragement really helped me to shape this report to perfection.

I express my heartfelt thanks to **Dr. K Ramesh**, Associate Professor & Project Coordinator, for his suggestions invaluable inputs and assessment really helped me to shape this report to perfection.

I wish to express my deep sense of gratitude to **Dr. Muni SekharVelpuru**, Associate Professor& Head, Department of Information Technology, Vardhaman College of Engineering, for his intense support and encouragement, which helped me to mold my project into a successful one.

I also owe my special thanks to our honourable Principal **Dr. J. V. R Ravindra**, of Vardhaman College of Engineering, for providing all the infrastructural facilities and congenial atmosphere to complete the project successfully.

I avail this opportunity to express my deep sense of gratitude and heartfelt thanks to **Dr. T. Vijender Reddy**, Chairman and **Sri T. Upender Reddy**, Secretary, of Vardhaman College of Engineering, for providing the infrastructural facilities and congenial atmosphere to complete the project successfully.

I also thank all the staff members of the Information Technology department for their valuable support and generous advice.

Finally, thanks to all my friends and family members for their continuous support and enthusiastic help.

M Soumya (18881A1242)

ABSTRACT

Handwritten Character Recognition (HCR) is ability of a Machine, automatically to detect, recognize and interpret handwritten characters from an Image. The automatic recognition of handwritten character is powerful tool in many applications where large volume of handwritten data has to process, example, recognition of addresses and postal codes on envelopes, interpretation of amounts on bank checks, and document analysis, etc. Pattern Recognition is extremely useful technology to recognize characters, but multi-variant, noise in input images reduced accuracy of character recognition In this project we are using deep learning techniques to classify the characters in image. For this we are training a Convolutional Neural Network (CNN) with all possible variants and noisy character data to the model which intern recognize multi-variant and noisy data with maximum accuracy. The output of CNN is given to Recurrent Neural Network (RNN) which gives sequence of characters as its output. This output is given to Connectionist Temporal Classification (CTC) to calculate the loss and decode the word. Thus the digital text is formed.

Keywords- HCR, Deep learning, CNN, Pattern Recognition, RNN, CTC.

TABLE OF CONTENTS

	Page No's
Title Page	I
Declaration	II
College Certificate	III
Internship Certificate	IV
Acknowledgement	V
Abstract	VI
Table of Contents	VII
List of Figures	VIII
List of Tables	IX
List of Screens	X
Symbols & Abbreviations	XI
1. INTRODUCTION	1-2
1.1 Introduction	1
1.2 Scope/ Problem definition	2
1.3 Purpose/ Objective of Project	2
1.4 Limitations of Project	2
2. LITERATURE SURVEY	3-4
2.1 Explain about existing system	3
2.2 Limitations of Existing System	3
2.3 Proposed Method and advantages	4
3. ANALYSIS	5-9
3.1 Introduction	5
3.2 Software Requirement Specification	5
3.2.1 User requirement	5-6
3.2.2 Software requirement	6
3.2.3 Hardware requirement	6
3.3 Algorithms and Flowcharts	6-9
4. DESIGN	10-12
4.1 Introduction	10
4.2 DFD / ER / UML diagram	
(Any other project diagrams-discuss with guide)	10-11
4.3 Module design and organization	12
5. IMPLEMENTATION	13-20
5.1 Introduction	13
5.2 Explanation of Key functions	13-16
5.3 Technology	16-17
5.4 Method of Implementation	17-20
5.2.1 Forms	17
5.2.2 Output Screens	18-20

5.2.3 Result Analysis	20
6. TESTING& RESULTS	21-23
6.1 Introduction	21
6.2 Design of test cases and scenarios	21-22
6.3 Validation	22-23
7. CONCLUSION	24
REFERENCES	25

LIST OF FIGURES

<u>FIGURE NO.</u>	<u>TITLE OF FIGURES</u>	<u>PAGE NO.</u>
1	Image taken from IAM dataset and its transcription into digital text	5
2	A CNN sequence to classify hand written characters	7
3	Recurrent Neural Networks	8
4	Connectionist Temporal Classification	9
5	Flow chart of HCR	9
6	ER diagram of HCR	10
7	Use case diagram of HCR	11
8	Sequence diagram of HCR	11
9	Proposal Approach	13
10	Implementation of HCR	13
11	The NN written as mathematical function	14
12	CNN output	18
13	RNN and CTC output	19
14	Analysis	20
15	Test cases and output	21-22

LIST OF TABLES

<u>TABLE NUMBER</u>	<u>TITLE OF TABLE</u>	<u>PAGE NO</u>
1	Modules and versions	6

LIST OF SCREENS

<u>SCREEN NUMBER</u>	<u>TITLE OF SCREEN</u>	<u>PAGE NO</u>
1	CNN conv2D layer	15
2	CNN relu and pooling	15
3	RNN LSTM	15
4	RNN bi-directional layer	15
5	Ground truth text encoded as sparse tensor	16
6	Calculating loss and decoding operation	16

ABBREVIATIONS

HCR: Handwritten Character Recognition

CNN: Convolutional Neural Network

RNN: Recurrent Neural Network

CTC: Connectionist Temporal Classification

TF: Tensor Flow

1. INTRODUCTION

1.1 INTRODUCTION

Despite the abundance of technological writing tools, many people still choose to take their notes traditionally: with pen and paper. However, there are drawbacks to hand writing text. It's difficult to store and access physical documents in an efficient manner, search through them efficiently and to share them with others. Thus, a lot of important knowledge gets lost or does not get reviewed because of the fact that documents never get transferred to digital format. Handwritten text is a very general term, and I wanted to narrow down the scope of the project by specifying the meaning of handwritten text for our purposes. In this project, I took on the challenge of classifying the image of any handwritten word, which might be of the form of cursive or block writing.

Even though, Text recognition in the handwritten documents has been studied as one of the prominent research areas by different researchers during the last few decades and because of that many automatic handwritten systems are developed by different researchers in past. However, the recognition algorithm and its efficiency is still an open research issue. Due to the vast inconsistency in handwriting styles, frequently the state-of-the-art handwriting recognition systems gets fail to provide satisfactory performance on various types of handwriting samples. Handwriting recognition has been one of the most fascinating and challenging research areas in field of image processing and pattern recognition in the recent years. It contributes immensely to the advancement of an automation process and can improve the interface between man and machine in numerous applications. Several research works have been focusing on new techniques and methods that would reduce the processing time while providing higher recognition accuracy.

Available approaches to handwriting recognition usually consist of various steps which mainly include 1. Pre-processing, 2. Feature extraction, 3. Classification 4. Post processing. However, feature extraction and classifier design are the two major steps of any recognition system. Many researchers made different type of handwritten text recognition systems for different languages such as English, Chinese, Arabic, Japanese, Bangla, Malyalam etc. Still the recognition problems of these scripts cannot be considered to be entirely solved.

1.2 SCOPE

Offline handwriting recognition, often referred to as optical character recognition, is performed after the writing is completed by converting the handwritten document into digital form. First, the handwriting to be recognized is digitized through scanner. Second it is send to Convolutional Neural Networks (CNN) layers, output of this is given to recurrent NN (RNN) layer as input and a final Connectionist Temporal Classification (CTC) layer.

1.3 OBJECTIVE OF PROJECT

Handwriting recognition (HWR), also known as **Handwritten Character Recognition (HCR)**, is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. The image of the written text may be sensed "off line" from a piece of paper by optical scanning. Alternatively, the movements of the pen tip may be sensed "on line", for example by a pen-based computer screen surface, a generally easier task as there are more clues available. A handwriting recognition system handles formatting, performs correct segmentation into characters, and finds the most plausible words.

The objective of this project is to identify handwritten characters with the use of neural networks. We have to construct suitable neural network and train it properly. The program should be able to extract the characters one by one and map the target output for training purpose. After automatic processing of the image, the training dataset has to be used to train “classification engine” for recognition purpose.

1.4 LIMITATIONS OF PROJECT

- Hand written Character Recognition (HCR) will recognize the text with good accuracy only for one word and gives less accuracy with the text containing sentences.

2. LITERATURE SURVEY

Optical Character Recognition

Optical Character Recognition is the first handwriting recognition techniques. Most of the scanning suits offer some form of the Optical Character Recognition. This form allows the users to scan the handwritten documents to be scan and it translates the words into basic text documents in the computer. That is how simple and easy it is. Optical Character Recognition is also used by some archivists. They use it as a way of converting large quantities of handwritten historical documents into an easy searchable text documents.

2.1 EXPLANATION OF EXISTING SYSTEM

An OCR system consists of a normal scanner and some special software. The scanner is used to scan text on a document or piece of paper into the computer. The OCR software then examines the page and changes the letters into a form that can be edited or processed by a normal word processing package. The ability to scan the characters accurately depends on how clear the writing is. Scanners have been improved to be able to read different styles and sizes of text as well as neat handwriting. Although they are often up to 95% accurate, any text scanned with OCR needs careful checking because some letters can be misread. OCR is used to automatically recognise postcodes on letters at sorting offices.

OCR is cheaper than paying someone to manually enter large amounts of text, much faster than someone manually entering large amounts of texts.

2.2 LIMITATIONS OF EXISTING SYSTEM

- Not 100% accurate, there are likely to be some mistakes made during the process.
- All documents need to be checked over carefully and then manually corrected.
- If the original document is of poor quality or the handwriting difficult to read, more mistakes will occur.
- Not worth doing for small amounts of text

2.3 PROPOSED METHOD AND ADVANTAGES

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.

Neural networks are able to learn features from analysing a dataset, and then classify an unseen image based on weights. Features are extracted in the convolutional layers, where a kernel is passed over the image to extract a certain feature. In the end result, multiple kernels learn all the features within a dataset, in order to make classifications.

As the input layer (and therefore also all the other layers) can be kept small for word-images, NN-training is feasible on the CPU (of course, a GPU would be better). This implementation is the bare minimum that is needed for HTR using TF.

We use a NN for our task. It consists of convolutional NN (CNN) layers, recurrent NN (RNN) layers and a final Connectionist Temporal Classification (CTC) layer.

The main advantage of neural network is cost and time benefit and high quality and accuracy in outputs. Neural networks are considered as trainable brains. You feed them information about your organization and train them in order to perform tasks such as report generation. These networks will use that new information, training, and work experience to improve and adapt in a similar way that a human worker learns. However, these networks are faster than the human workforce and function at a rapid pace.

3. ANALYSIS

3.1 INTRODUCTION

Handwritten text is a very general term, and we wanted to narrow down the scope of the project by specifying the meaning of handwritten text for our purposes. In this project, we took on the challenge of classifying the image of any handwritten word, which might be of the form of cursive or block writing.

This project can be combined with algorithms that segment the word images in a given line image, which can in turn be combined with algorithms that segment the line images in a given image of a whole handwritten page. With these added layers, our project can take the form of a deliverable that would be used by an end user, and would be a fully functional model that would help the user solve the problem of converting handwritten documents into digital format, by prompting the user to take a picture of a page of notes. Note that even though there needs to be some added layers on top of our model to create a fully functional deliverable for an end user, we believe that the most interesting and challenging part of this problem is the classification part, which is why we decided to tackle that instead of segmentation of lines into words, documents into lines, etc.

We approach this problem with complete word images because CNNs tend to work better on raw input pixels rather than features or parts of an image. Given our findings using entire word images, we sought improvement by extracting characters from each word image and then classifying each character independently to reconstruct a whole word. In summary, in both of our techniques, our models take in an image of a word and output the name of the word.

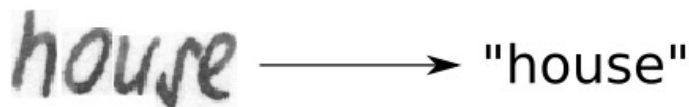


Fig 1 Image of word (taken from IAM) and its transcription into digital text.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

3.2.1 User requirement

The User Requirements Specification describes the business needs for what users require from the system.

1. The developed system should recognize handwritten English character present in the image.
2. System must provide the quality of service to user.
3. System must provide accuracy for character recognition.

3.2.2 Software requirement

- Jupyter notebook
- Python 3

Table 1 Module and versions

module	Version
absl-py	0.7.0
astor	0.7.1
editdistance	0.5.2
gast	0.2.2
grpcio	1.18.0
h5py	2.9.0
Keras-Applications	1.0.7
Keras-Preprocessing	1.0.8
Markdown	3.0.1
opencv-python	4.0.0.21
numpy	1.16.1
protobuf	3.6.1
six	1.12.0
tensorboard	1.12.2
tensorflow	1.12.0
termcolor	1.1.0
Werkzeug	0.14.1

3.2.3 Hardware requirement

- Intel i3 Processor
- 128 MB RAM
- 10GB Hard Disk

3.3 ALGORITHMS AND FLOWCHARTS

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.

Convolutional Neural Network (CNN)

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which 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. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNet have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

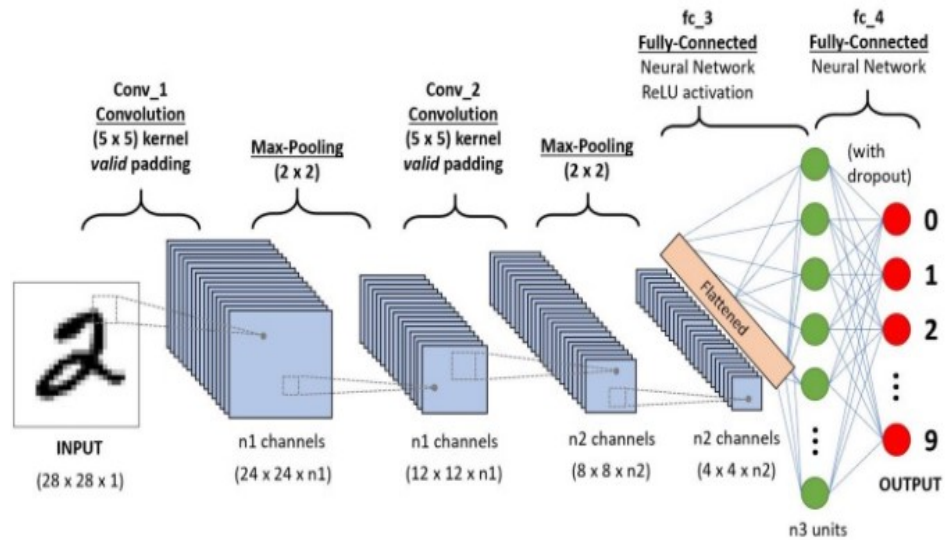


Fig 2 A CNN sequence to classify handwritten characters

Recurrent Neural Network (RNN)

RNNs are a powerful and robust type of neural network, and belong to the most promising algorithms in use because it is the only one with an internal memory. Because of their internal memory, RNN's can remember important things about the input they received, which allows them to be very precise in predicting what's coming next. This is why they're the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more. Recurrent neural networks can form a much deeper understanding of a sequence and its context compared to other algorithms. In a RNN the information cycles through

a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously. A usual RNN has a short-term memory. In combination with a LSTM they also have a long-term memory.

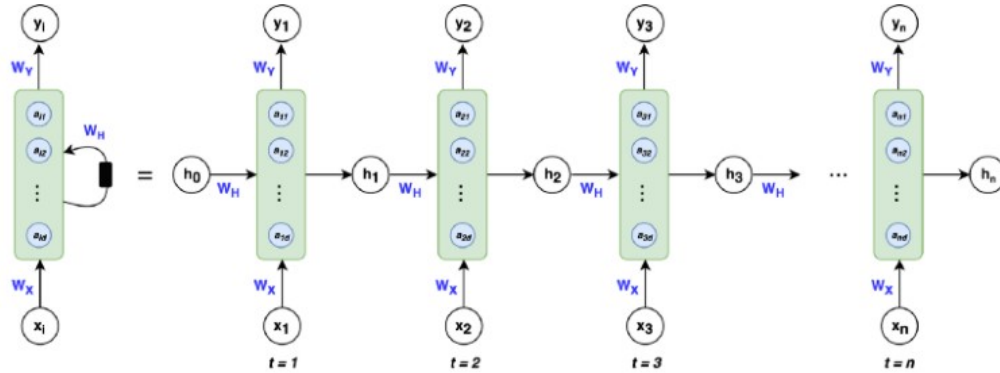


Fig 3 Recurrent Neural Network

Connectionist Temporal Classification (CTC)

Connectionist temporal classification (CTC) is a type of neural network output and associated scoring function, for training recurrent neural networks (RNNs) such as LSTM networks to tackle sequence problems where the timing is variable. It can be used for tasks like on-line handwriting recognition or recognizing phonemes in speech audio. CTC refers to the outputs and scoring, and is independent of the underlying neural network structure.

The input is a sequence of observations, and the outputs are a sequence of labels, which can include blank outputs. The difficulty of training comes from there being many more observations than there are labels. For example, in speech audio there can be multiple times slices which correspond to a single phoneme. Since we don't know the alignment of the observed sequence with the target labels we predict a probability distribution at each time step. A CTC network has a continuous output (e.g. softmax), which is fitted through training to model the probability of a label. CTC does not attempt to learn boundaries and timings: Label sequences are considered equivalent if they differ only in alignment, ignoring blanks. CTC scores can then be used with the back-propagation algorithm to update the neural network weights.

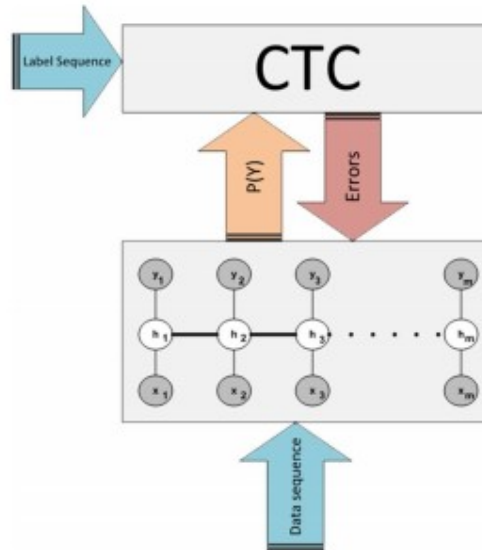


Fig 4 Connectionist Temporal Classification

Flowchart

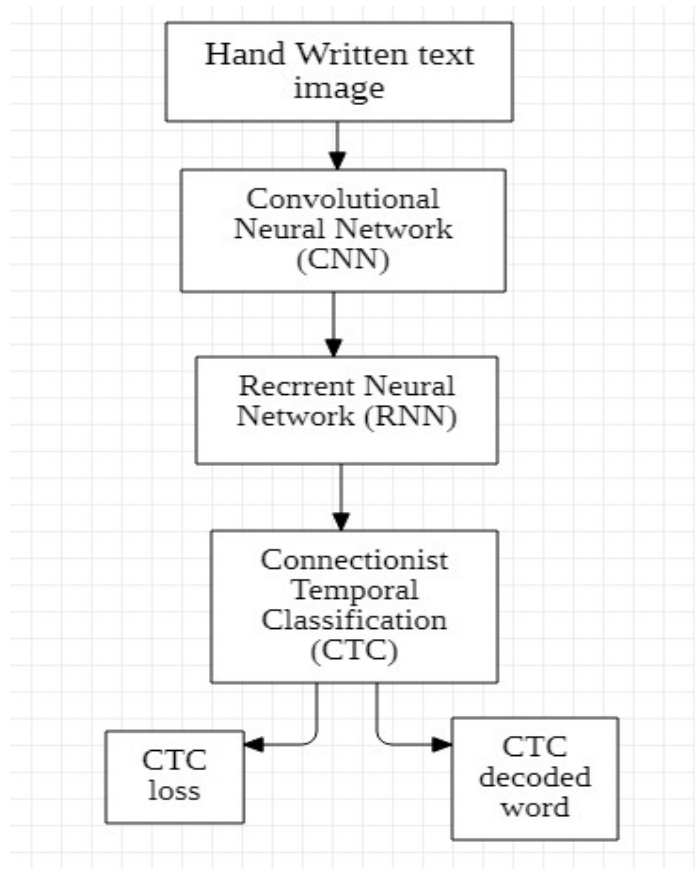


Fig 5 Flowchart of HCR

4. DESIGN

4.1 INTRODUCTION

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.

ER diagram

Entities here are user, image, textual region, text recognition system, character recognition. User captures the image which is further processed by text recognition system and localized by character recognition and finally converted to text. Image captured by user contains textual regions which has lines, words.

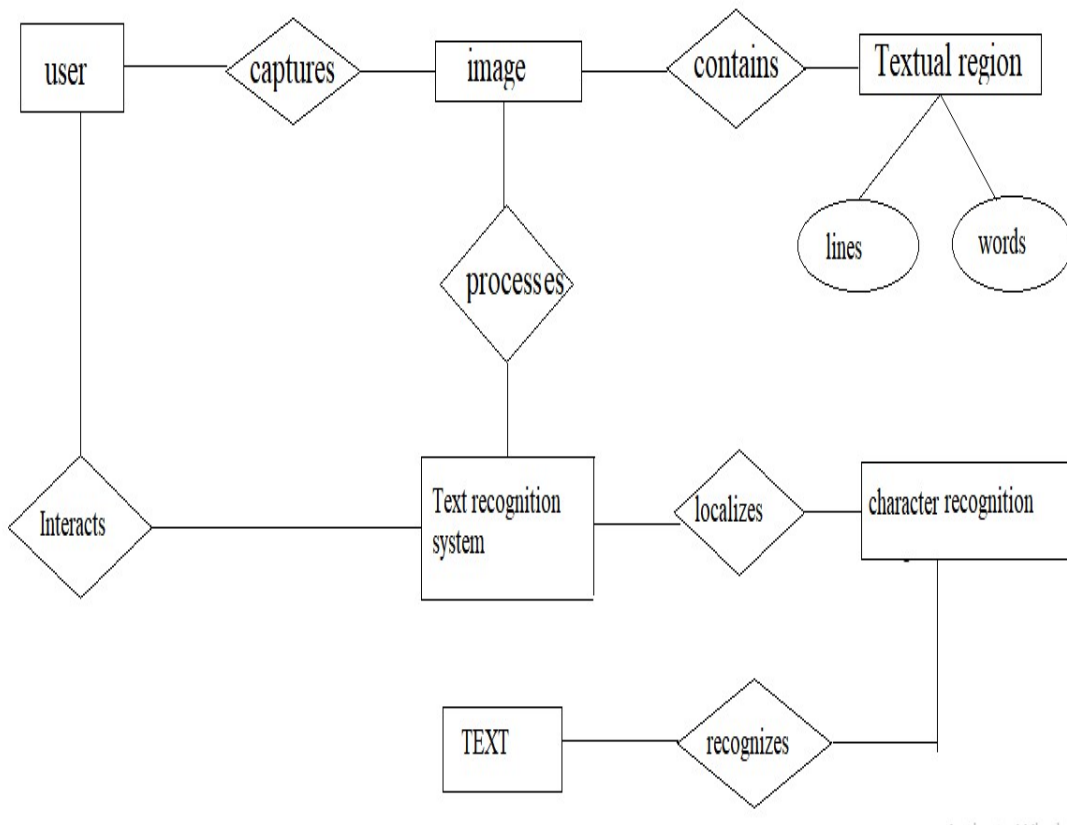


Fig 6 ER diagram of HCR

UML diagrams

Use case diagram

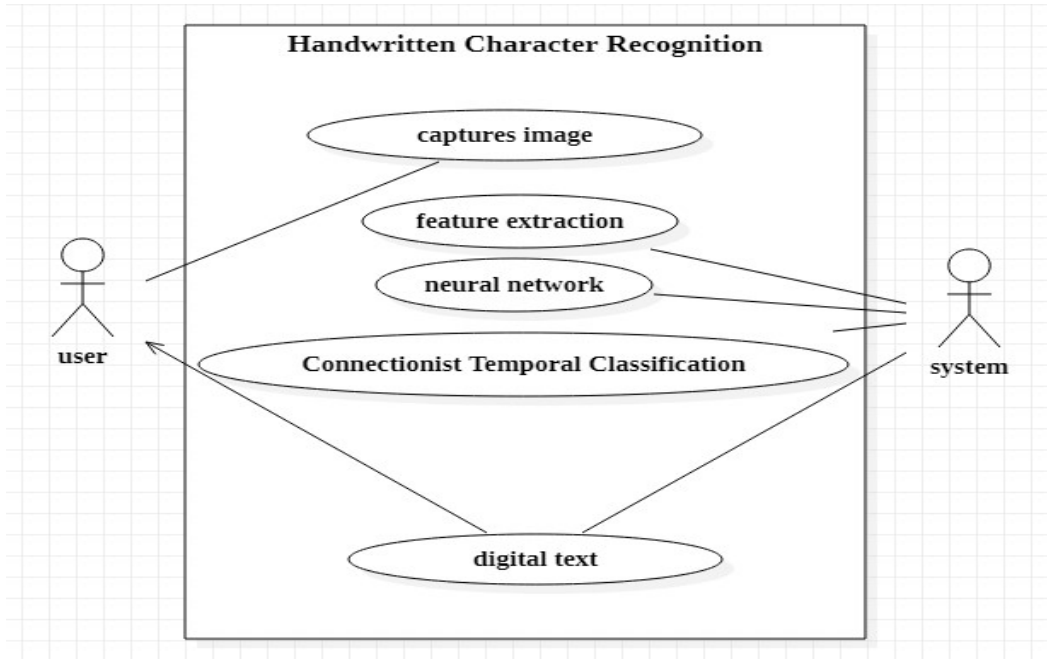


Fig 7 Use case diagram of HCR

Sequence diagram

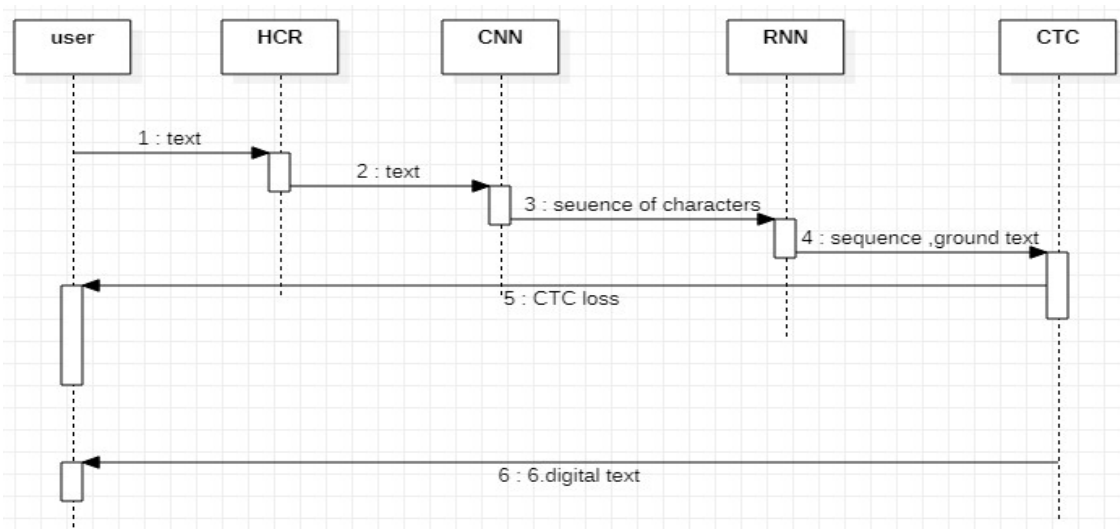


Fig 8 sequence diagram of HCR

4.3 MODULE DESIGN AND ORGANIZATION

Numpy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Tensor Flow: Tensor Flow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google.

OpenCV: OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

5. IMPLEMENTATION

5.1 INTRODUCTION

Handwritten Character Recognition (HCR) system implemented with Tensor Flow (TF) and trained on the IAM off-line HTR dataset. This Neural Network (NN) model recognizes the text contained in the images of segmented words as shown in the illustration below. As these word-images are smaller than images of complete text-lines, the NN can be kept small and training on the CPU is feasible. 3/4 of the words from the validation-set are correctly recognized and the character error rate is around 10%.

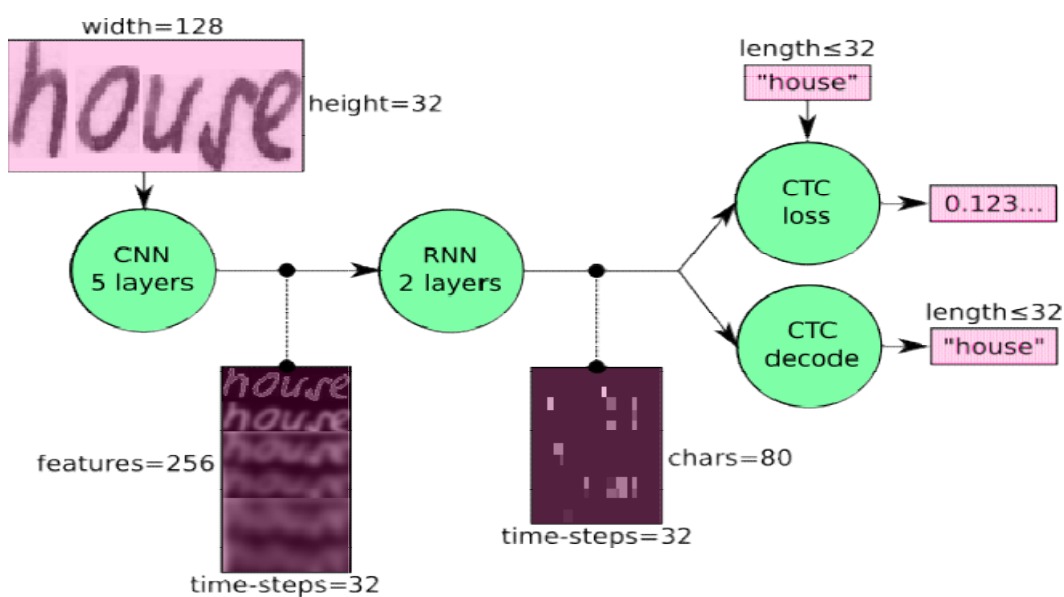


Fig 9 Proposal approach

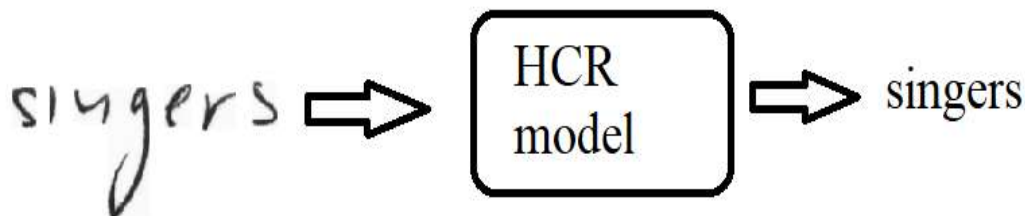


Fig 10 Implementation of HCR

5.2 EXPLANATION OF KEY FUNCTIONS

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 $W \times H$ to a character sequence (c_1, c_2, \dots) 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).

$$\text{NN: } \underset{W \times H}{M} \rightarrow \underset{0 \leq n \leq L}{(c_1, c_2, \dots, c_n)}$$

Fig 11 The NN written as mathematical function which maps an image M to character sequence (c_1, c_2)

Implementation using Tensor Flow (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, puts 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 $k \times k$ to be used in the convolution operation.

```
kernel = tf.Variable(tf.truncated_normal([k, k, chIn, chOut], stddev=0.1))
conv = tf.nn.conv2d(inputTensor, kernel, padding='SAME', strides=(1, 1, 1, 1))
```

Output Screen 1 CNN conv2d layer

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

```
relu = tf.nn.relu(conv)
pool = tf.nn.max_pool(relu, [1, px, py, 1], [1, sx, sy, 1], 'VALID')
```

Output Screen 2 CNN relu and pooling

These steps are repeated for all layers in a for-loop RNN Create and stack two RNN layers with 256 units each.

```
cells = [tf.contrib.rnn.LSTMCell(num_units=256, state_is_tuple=True) for _ in range(2)]
stacked = tf.contrib.rnn.MultiRNNCell(cells, state_is_tuple=True)
```

Output Screen3 RNN LSTM

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 forward and backward 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.

```
((fw, bw),_) = tf.nn.bidirectional_dynamic_rnn(cell_fw=stacked, cell_bw=stacked, inputs=inputTensor, dtype=inputTensor.dtype)
```

Output Screen 4RNN bi directional 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.

```
gtTexts = tf.SparseTensor(tf.placeholder(tf.int64, shape=[None, 2]), tf.placeholder(tf.int32, [None]), tf.placeholder(tf.int64, [2]))
seqLen = tf.placeholder(tf.int32, [None])
```

Output Screen 5 Ground truth text encoded as sparse tensor

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

```
loss = tf.nn.ctc_loss(labels=gtTexts, inputs=inputTensor, sequence_length=seqLen, ctc_merge_repeated=True)
decoder = tf.nn.ctc_greedy_decoder(inputs=inputTensor, sequence_length=seqLen)
```

Output Screen6 calculating loss and decoding operation

5.3 TECHNOLOGY

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, which applies a filter kernel of size 5×5 in the first two layers and 3×3 in the last three layers to the input. Then, the non-linear RELU function is applied. Finally, a pooling layer summarizes image regions and outputs a downsized version of the input. While the image height is downsized by 2 in each layer, feature maps (channels) are added, so that the output feature map (or sequence) has a size of 32×256 .

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

CTC: while training the NN, the CTC is given the RNN output matrix and the ground truth text and it computes the loss value. While inferring, the CTC is only given the matrix and it decodes it

into the final text. Both the ground truth text and the recognized text can be at most 32 characters long.

5.4 METHOD OF IMPLEMENTATION

5.4.1 Forms

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.

The IAM Handwriting Database is hierarchically structured into forms (The name of the files correspond to the naming scheme of the LOB Corpus)

The data-loader expects the IAM dataset (or any other dataset that is compatible with it) in the data/ directory. Follow these instructions to get the dataset:

1. Register for free at this website.
 2. Download words/words.tgz.
 3. Download ascii/words.txt.
 4. Put words.txt into the data/ directory.
 5. Create the directory data/words/.
 6. Put the content (directories a01, a02, ...) of words.tgz into data/words/.
 7. Go to data/ and run python checkDirs.py for a rough check if everything is ok.
- Data:
 - data/ascii.tgz - Contains summarized meta information in ascii format.
 - data/formsA-D.tgz data/formsE-H.tgz data/formsI-Z.tgz - Contains form images (example: a01-122.png).
 - data/lines.tgz - Contains text lines (example: a01/a01-122/a01-122-02.png).
 - data/sentences.tgz - Contains sentences, one for each line (example: a01/a01-122/a01-122-s01-02.png).
 - data/words.tgz - Contains words (example: a01/a01-122/a01-122-s01-02.png).
 - data/xml.tgz - Contains the meta-information in XML format (example: a01-122.xml).

You can look at the images in each directory or download a compressed archive in each top level directory.

5.4.2 Output Screens

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: Fig. 12 shows the output of the CNN layers which is a sequence of length 32. Each entry contains 256 features. Of course, these features are further processed by the RNN layers, however, some features already show a high correlation with certain high-level properties of the input image: there are features which have a high correlation with characters (e.g. “e”), or with duplicate characters (e.g. “tt”), or with character-properties such as loops (as contained in handwritten “l” s or “e” s).

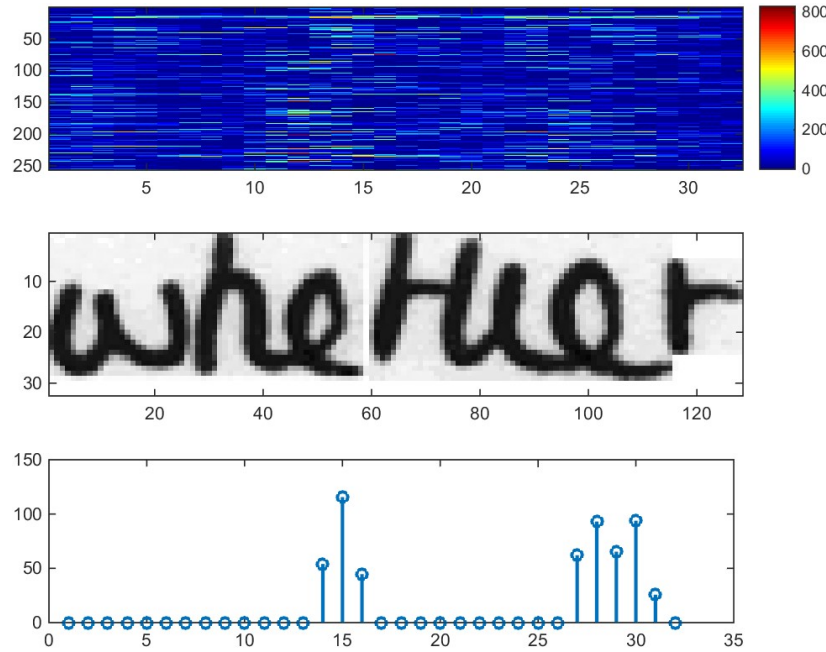


Fig 12 Top 256 feature per time step are computed by CNN layers. Middle: input image Bottom: plot of the 32nd feature, which has a high correlation with the occurrence of the character “e” in the image

RNN output: Figure 13 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.

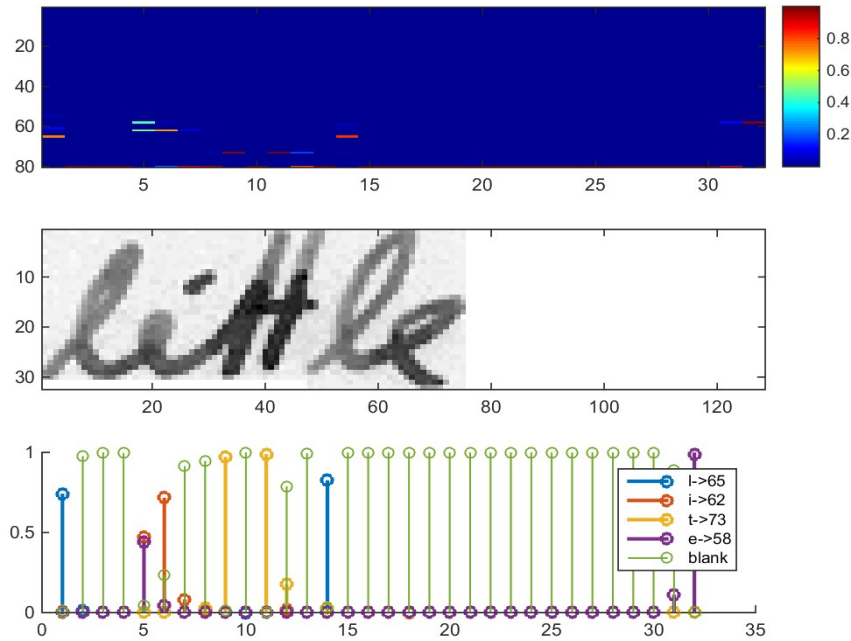


Fig 13 Top: output matrix of the RNN layers. Middle: input image. Bottom: probabilities for the characters “l”, “i”, “t”, “t”, “l”, “e” and the CTC blank label

Connectionist Temporal Classification (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.
- 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” →

“l---i--t-t--l-...-e” →

“little”.

5.4.3 Result Analysis

Run python analyze.py with the following arguments to analyze the image file data/analyze.png with the ground-truth text "are":

- --relevance: compute the pixel relevance for the correct prediction.
- --invariance: check if the model is invariant to horizontal translations of the text.
- No argument provided: show the results.

Results are shown in the plots below. The pixel relevance (left) shows how a pixel influences the score for the correct class. Red pixels vote for the correct class, while blue pixels vote against the correct class. It can be seen that the white space above vertical lines in images is important for the classifier to decide against the "i" character with its superscript dot. Draw a dot above the "a" (red region in plot) and you will get "aive" instead of "are".

The second plot (right) shows how the probability of the ground-truth text changes when the text is shifted to the right. As can be seen, the model is not translation invariant, as all training images from IAM are left-aligned. Adding data augmentation which uses random text-alignments can improve the translation invariance of the model.

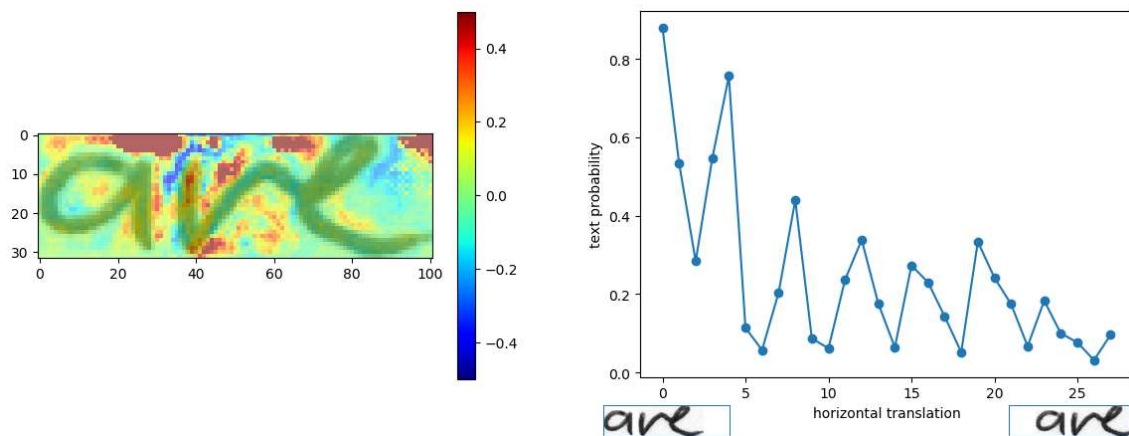


Fig 14 analysis

6 TESTING& RESULTS

6.1 INTRODUCTION

The model is a stripped-down version of the HTR system I implemented for my thesis. What remains is what I think is the bare minimum to recognize text with an acceptable accuracy. The implementation only depends on numpy, cv2 and tensorflow imports. It consists of 5 CNN layers, 2 RNN (LSTM) layers and the CTC loss and decoding layer.

- The input image is a gray-value image and has a size of 128x32
- 5 CNN layers map the input image to a feature sequence of size 32x256
- 2 LSTM layers with 256 units propagate information through the sequence and map the sequence to a matrix of size 32x80. Each matrix-element represents a score for one of the 80 characters at one of the 32 time-steps
- The CTC layer either calculates the loss value given the matrix and the ground-truth text (when training), or it decodes the matrix to the final text with best path decoding or beam search decoding (when inferring)
- Batch size is set to 50

6.2 DESIGN OF TEST CASES AND SCENARIOS



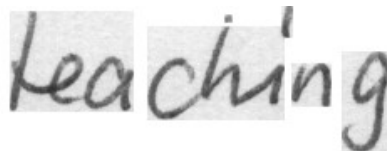
Recognized: "North"
Probability: 0.9764741



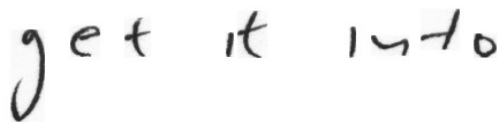
Recognized: ")"
Probability: 0.9315802



Recognized: "little"
Probability: 0.96625465



Recognized: "teaching"
Probability: 0.92382056



Recognized: "gento"
Probability: 0.42871612

Fig 15 test cases and output

6.3 VALIDATION

If you want to train the model from scratch, delete the files contained in the model/ directory. Otherwise, the parameters are loaded from the last model-snapshot before training begins. Then, go to the src/ directory and execute `python main.py --train`. After each epoch of training, validation is done on a validation set (the dataset is split into 95% of the samples used for training and 5% for validation as defined in the class `DataLoader`). If you only want to do

validation given a trained NN, execute `python main.py --validate`. Training on the CPU takes 18 hours on my system (VM, Ubuntu 16.04, 8GB of RAM and 4 cores running at 3.9GHz). The expected output is shown below.

```
> python main.py --train
```

```
Init with new values
```

```
Epoch: 1
```

```
Train NN
```

```
Batch: 1 / 500 Loss: 130.354
```

```
Batch: 2 / 500 Loss: 66.6619
```

```
Batch: 3 / 500 Loss: 36.0154
```

```
Batch: 4 / 500 Loss: 24.5898
```

```
Batch: 5 / 500 Loss: 20.1845
```

```
Batch: 6 / 500 Loss: 19.2857
```

```
Batch: 7 / 500 Loss: 18.3493
```

```
...
```

```
Validate NN
```

```
Batch: 1 / 115
```

```
Ground truth -> Recognized
```

```
[OK] "," -> ","
```

```
[ERR:1] "Di" -> "D"
```

```
[OK] "," -> ","
```

```
[OK] "" -> ""
```

```
[OK] "he" -> "he"
```

```
[OK] "told" -> "told"
```

```
[ERR:2] "her" -> "nor"
```

```
...
```

```
Character error rate: 13.956289%. Word accuracy: 67.721739%.
```

7. CONCLUSION

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.

Future Enhancement

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.

To improve the recognition accuracy, following developments can be done:

- Data augmentation: increase dataset-size by applying further (random) transformations to the input images
- Remove cursive writing style in the input images.
- Increase input size (if input of NN is large enough, complete text-lines can be used)
- Add more CNN layers
- Replace LSTM by 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

References

- [1] Hong-Phuong Tran, Andrew Smith, Eric Dimla, "Offline Handwritten Text Recognition using Convolutional Recurrent Neural Network", *Advanced Computing and Applications (ACOMP) 2019 International Conference on*.
- [2] <https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5>
- [3] <https://github.com/githubharald/SimpleHTR>
- [4] <https://fki.tic.heia-fr.ch/databases/iam-handwriting-database>
- [5] Li Chen, Song Wang, Wei Fan, Jun Sun and Satoshi Naoi, "Beyond human recognition: A CNN-based framework for handwritten character recognition", *3rd LAPR Asian Conference on Pattern Recognition*, ISSN 2327-0985
- [6] H. I. Avi-Itzhak, T. A. Diep and H. Garland, "High accuracy optical character recognition using neural networks", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18,