DAYANANDA SAGAR UNIVERSITY

KUDLU GATE, BANGALORE – 560068

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

Special Topic- II (19CS3604) Report

on

Turning Handwritten Documents into Digitized Versions

By

Arpit Kumar (ENG19CS0042)
Mishra Ashutosh Pradeep (ENG19CS0050)
B Vijay Kumar Kakra (ENG19CS0059)
Dheeraj Kumar (ENG19CS0085)

Under the supervision of
Prof. Nazmin Begum
Asst.Professor CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF ENGINEERING

DAYANANDA SAGAR UNIVERSITY,

(2021-2022)

DAYANANDA SAGAR UNIVERSITY

KUDLU GATE, BANGALORE – 560068

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **Arpit Kumar, Mishra Ashutosh Pradeep, B Vijay Kumar Kakra, Dheeraj Kumar** bearing USN **ENG19CS0042, ENG19CS0050, ENG19CS0059, ENG19CS0085** has satisfactorily completed his Special Topics II as prescribed by the University for the 6th semester B.Tech Programme in Computer Science & Engineering for Software Engineering & Project Management course during the year 2021-22 at the School of Engineering, Dayananda Sagar University, Bangalore**.**

Date:______________

Signature of faculty in-charge

Max Marks	Marks Obtained

Chairman

Department of Computer Science & Engineering

DECLARATION

I hereby declare that the work presented in this minor project entitled - "Turning Handwritten Documents into Digitized version", has been carried out by me and it has not been submitted for the award of any degree, diploma or the Special Topics II of any other collegeor university.

Arpit Kumar (ENG19CS0042)
Mishra Ashutosh Pradeep (ENG19CS0050)
B Vijay Kumar Kakra (ENG19CS0059)
Dheeraj Kumar (ENG19CS0085)

ACKNOWLEDGEMENT

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

I am especially thankful to my Dean **Dr. Srinivas A** & Chairman **Dr. Girisha G S** Department of Computer Science & Engineering, for providing necessary departmentalfacilities, moral support and encouragement.

I am very much thankful to **Prof. Nazmin Begum**, Department of Computer Science & Engineering, for providing help and suggestions in completion of this Special Topics II successfully.

I have received a great deal of guidance and co-operation from my friends and I wish to thank all that have directly or indirectly helped us in the successful completion of this project work.

Arpit Kumar (ENG19CS0042)
Mishra Ashutosh Pradeep (ENG19CS0050)
B Vijay Kumar Kakra (ENG19CS0059)
Dheeraj Kumar (ENG19CS0085)

ABSTRACT

The project converts handwritten document into its corresponding digital format.

Since, digital documents can be easily manipulated, stored and retrieved, this project serves various applications such as processing cheques in banks, converting handwritten books into digital copies that can used for publication, retrieving information from application forms, etc.

This Project aims to give a new solution to traditional handwriting recognition techniques using concepts of Deep learning, Computer Vision, Regional Convolutional Neural Network(R-CNN) and Convolutional Neural Networks (CNN).

TABLE OF CONTENTS

TABLE OF FIGURE

CHAPTER 1

INTRODUCTION

Even though we live in a digital world, where we have and abundance of technological writing tools, many people still prefer taking notes traditionally with pen and paper. Therefore, handwritten text plays a major role.

So, there is a way to move towards digitized version using Deep Learning, Supervised learning, CNN, RCNN, CTC and ML

This method can be used for visualizing Doctor Prescription which is quite hard to read, scan and get digital form of our writing.

Example: - Google Lens

1.1 ABOUT THE PROJECT

The process of converting the handwritten document into the digital format requires several processes. Firstly, the handwritten document is scanned and converted to a digital image, which is preprocessed to remove the noise and artifacts in the image. OpenCV is used to detect and segment only the handwritten text into individual handwritten characters. Further, the position of each character relative to the original document is stored. Once image is processed, each character image is passed one-by-one to the built and trained Convolutional Neural Network (CNN) model. The CNN model results a set of probabilities for each character class. The highest probability character class is chosen as the corresponding output fer the given input character. The output character is stored in a text document at its respective positions. Each word in the output text document verified using a spellcheck tool that corrects the incorrect words.

1.2 SCOPE

The system can be further enhanced in the future that overcomes all the drawbacks of the handwritten notes which are converted into digitized version. In future we are planning to extend this study to a larger extent where different embedding models can be considered on large variety of the datasets. The future is completely based on technology no one will use the paper and pen for writing. In that scenario they used write on touch pads so the inbuilt software which can automatically detects text which they writing and convert into digital text so that the searching and understanding very much simplified.

CHAPTER 2

PROBLEM DEFINITION

Computers and phones may be more ubiquitous than ever, but many people still prefer the traditional feeling of writing with ink on paper.

After all, this method served us well for hundreds of years of human history. Despite the availability of various technological writing tools, many people still choose to take their notes traditionally: with pen and paper. However, there are certain pitfalls in traditional way of handwritten text.

It is difficult to store and access physical documents in an organized manner, search through them efficiently and to share them with others.

Thus, handwriting recognition is the ability to interpret intelligible handwritten input from sources such as paper documents, touch-screens and other devices into digital form.

The hand transcription can easily identify by humans. Different languages have different patterns to spot. Humans can identify the text accurately. The hand transcription cannot be identified by the machine.

CHAPTER 3
LITERATURE SURVEY

[1] Mor, S. S., Solanki, S., Gupta, S., Dhingra, S., Jain, M., & Saxena, R. (2019). Handwritten text recognition: with deep learning and android. *International Journal of Engineering and Advanced Technology*, *8*(3S), 819-825.

Description- This research paper offers a new solution to traditional handwriting recognition techniques using concepts of Deep learning and computer vision. An extension of MNIST digits dataset called the Emnist dataset has been used. It contains 62 classes with 0-9 digits and A-Z characters in both uppercase and lowercase. An application for Android, to detect handwritten text and convert it into digital form using Convolutional Neural Networks, abbreviated as CNN, for text classification and detection, has been created. Prior to that we pre-processed the dataset and applied various filters over it.

[2] Manchala, S. Y., Kinthali, J., Kotha, K., Kumar, J. J. K. S., & Jayalaxmi, J. (2020). Handwritten text recognition using deep learning with Tensorflow. *International Journal of Engineering and Technical Research*, *9*(5).

Description- The scanned image from the dataset are taken to build Neural Network for training. CNN, RNN and CTC techniques is used for conversion with 90.3 accuracy

[3]Balci, B., Saadati, D., & Shiferaw, D. (2017). Handwritten text recognition using deep learning. *CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University, Course Project Report, Spring*, 752-759.

Description- Here Text image from database are taken for enhancement then after segmentation is done text image goes for classification, very basic steps are followed.

[4]MOHAN KRISHNA, B., SWATHI, M., PRAKASH, H., & RAHUL, N. (2018). Handwritten Document Conversion Using Regional Convolutional Neural Network.

Description- Project helps in overcoming the drawbacks of handwritten documents by converting them into the digital format using ML and NN

[5]Aqab, S., & Tariq, M. U. (2020). Handwriting recognition using artificial intelligence neural network and image processing. *International Journal of Advanced Computer and Application (IJACSA)*, *11*(7), 137-146.

Description- Aim of this research was to develop a system that will help in the classification and recognition of Handwriting characters and digits. The work of the current research can be extended for character recognition in other languages.

[6]Hemanth, G. R., Jayasree, M., Venii, S. K., Akshaya, P., & Saranya, R. (2021). CNN-RNN Based Handwritten Text Recognition. *ICTACT Journal on Soft Computing*, *12*(1), 2457-2463.

Description- An adaptive method is proposed for offline paragraph recognition by pre-processing and training the dataset consecutively with CNN and RNN. The input paragraph images are first pre-processed by using OpenCV contour techniques and are split into line images and further line images are processed into word images which are fed into the NN model layers during recognition.

[7]Obaid, A. M., El Bakry, H. M., Eldosuky, M. A., & Shehab, A. I. (2016). Handwritten text recognition system based on neural network. *Int. J. Adv. Res. Comput. Sci. Technol*, *4*(1), 72-77.

Description- A comparison with related work has been presented. ANNs have been trained for this purpose with various types of input samples and that’s why the developed program has an ability to test and classify the input character into 52 different classes with an accuracy of more than 95%. Two different learning algorithms have been used.

[8]Patel, D. K., Som, T., Yadav, S. K., & Singh, M. K. (2012). Handwritten character recognition using multiresolution technique and Euclidean distance metric.

Description- The results showing the average recognition accuracy, here it is 90%. When average recognition accuracy is optimal (maximum) for a particular level of multiresolution and appropriate resolution of character images, here it is 90%.

CHAPTER 4
PROJECT DESCRIPTION

PROJECT DESIGN

DATA FLOW DIAGRAM: -

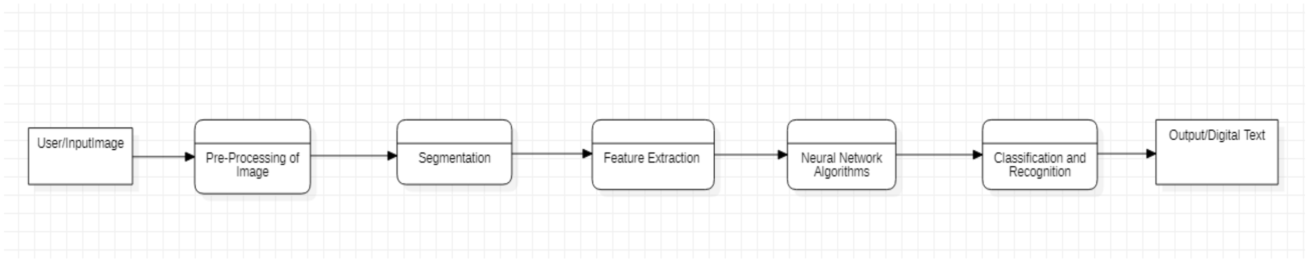


Figure 1. Data Flow Diagram

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored.

The flow of data in the model can be seen in 7 major steps which include

1. Input Image: -A handwritten text image is given as input to the system.
2. Pre-processing of image: -It is the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models.
3. Segmentation: - Here, the task of clustering parts of an image together that belong to the same object class is done.
4. Feature Extraction: -Includes convolution layer piles and sets of pooling layers to extract the feature from the given/Inputimage.
5. Neural Network Algorithms: -Applying the required NN Algorithms i.e., CNN, RNN, CTC
6. Classification and Recognition: - Here we check the produced image text with the actual text present in the input/given image.(or present in the database)
7. Output Image: -To display the output or the text present in the input image with atleast accuracy 80% or above 80%.

USE CASE DIAGRAM: -

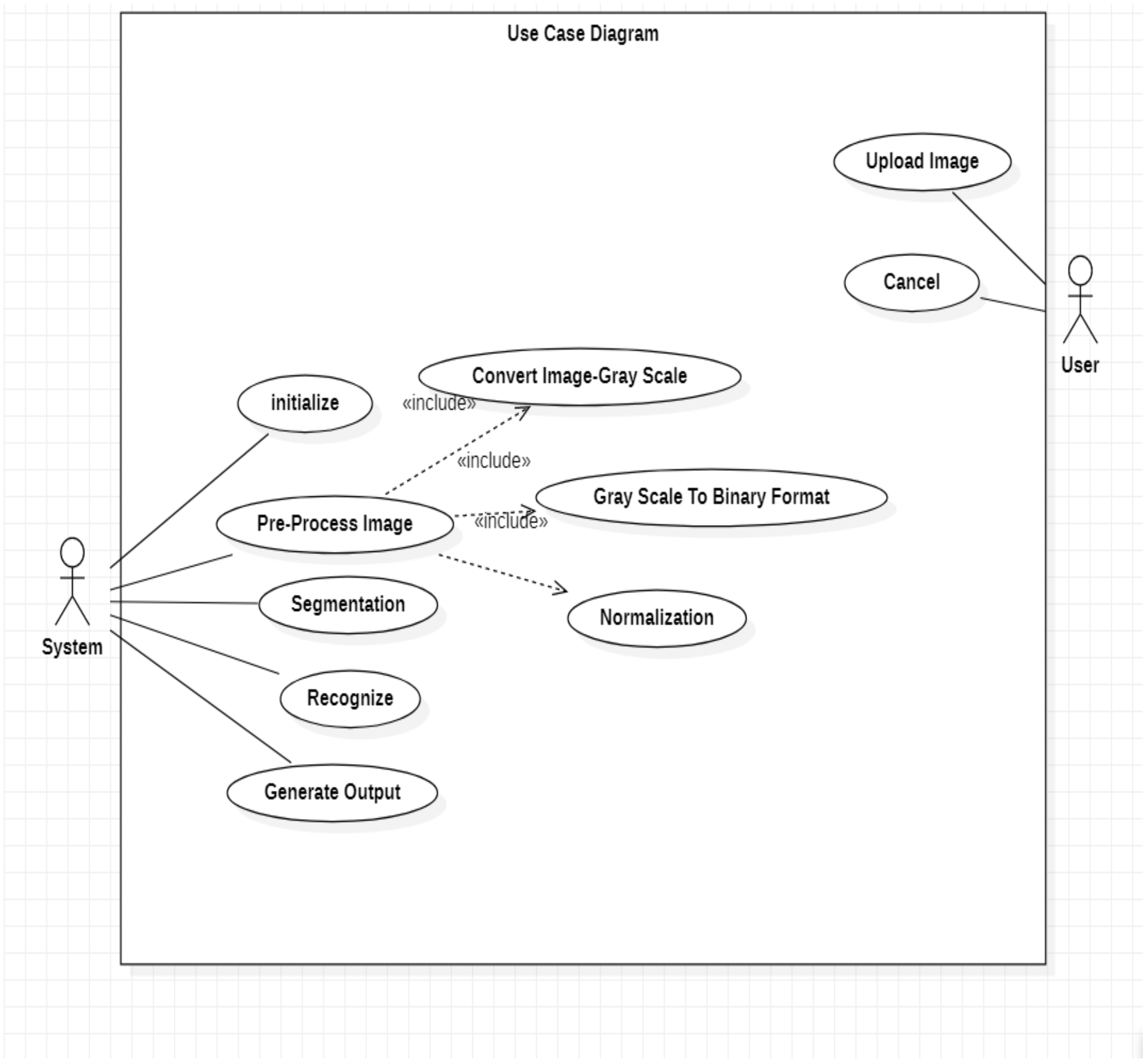


Figure 2. Use Case Diagram

SEQUENCE DIAGRAM: -

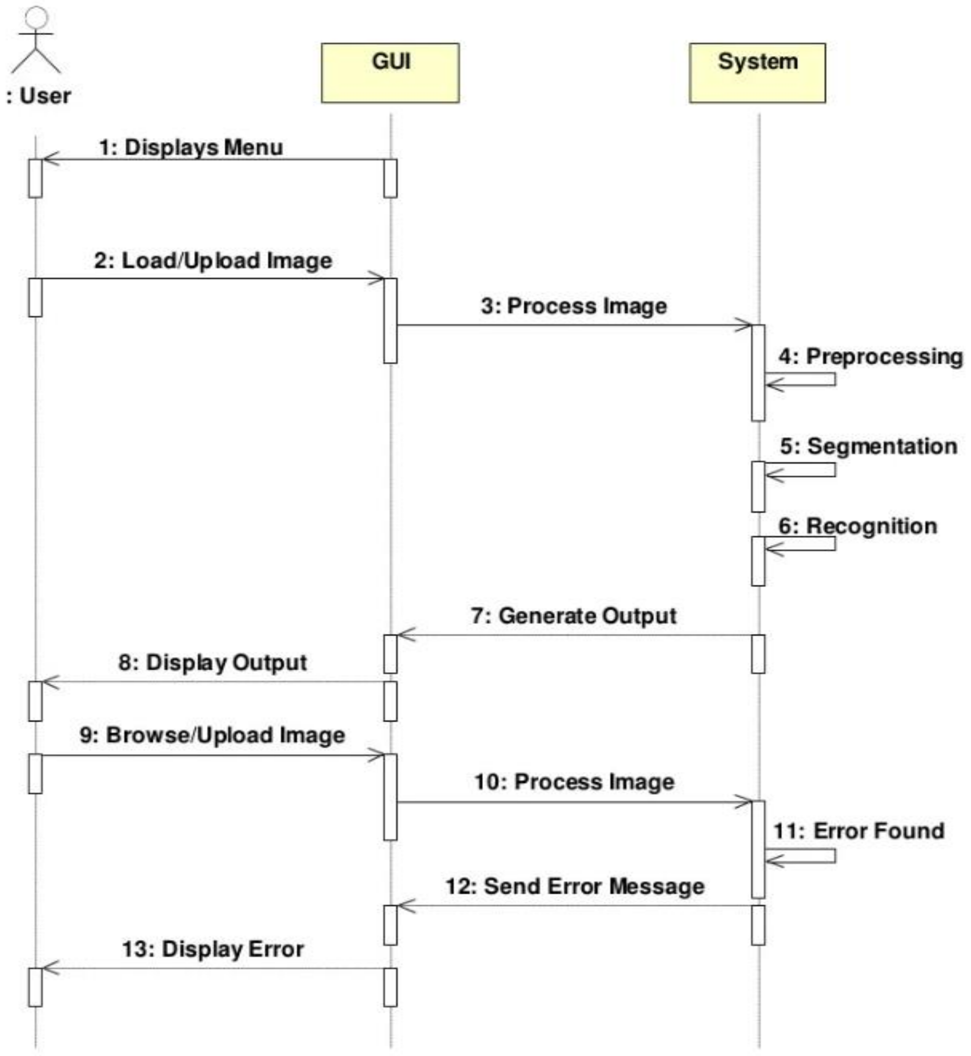


Figure 3. Sequence Diagram

CHAPTER 5

REQUIREMENTS

FUNCTIONAL REQUIREMENTS: -

The system should process the input given by the user only if it is an image file (JPG, PNG etc.)

System shall show the error message to the user when the input given is not in the required format.

System should detect characters present in the image.

System should retrieve characters in the image and display them to the user.

NON-FUNCTIONAL REQUIREMENTS: -

Performance: Handwritten characters in the input image will be recognized with an accuracy of about 80% and more.

Functionality: This software will deliver on the functional requirements mentioned in this document.

Availability: This system will retrieve the handwritten text regions only if the image contains written text in it.

Flexibility: It provides the user to load the image easily.

Learn ability: The software is very easy to use and reduces the learning work.

Reliability: This software will work reliably for low resolution images and not for graphical images.

HARDWARE AND SOFTWARE REQUIREMENTS: -

Software Requirement:

● Operating System: Microsoft Windows 10 pro.

● IDE: Visual Studio Code

● Browser: Google Chrome, Mozilla Firefox, or Microsoft Edge.

Hardware Requirements:

● Processor: Intel(R)Core (TM) i5-5300U CPU @2.30GHz

● Memory: Powerful enough to run the program.

● Ethernet Connection (LAN) or WIFI.

CHAPTER 6

METHODOLOGY

Project consists of Three steps:

1. Multi-scale feature Extraction → Convolutional Neural Network 5 Layers

2. Sequence Labeling (BLSTM-CTC) → Recurrent Neural Network (2 layers of LSTM) with CTC

3. Transcription → Decoding the output of the RNN (CTC decode)

Convolutional Neural Networks:

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, ConvNets 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.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

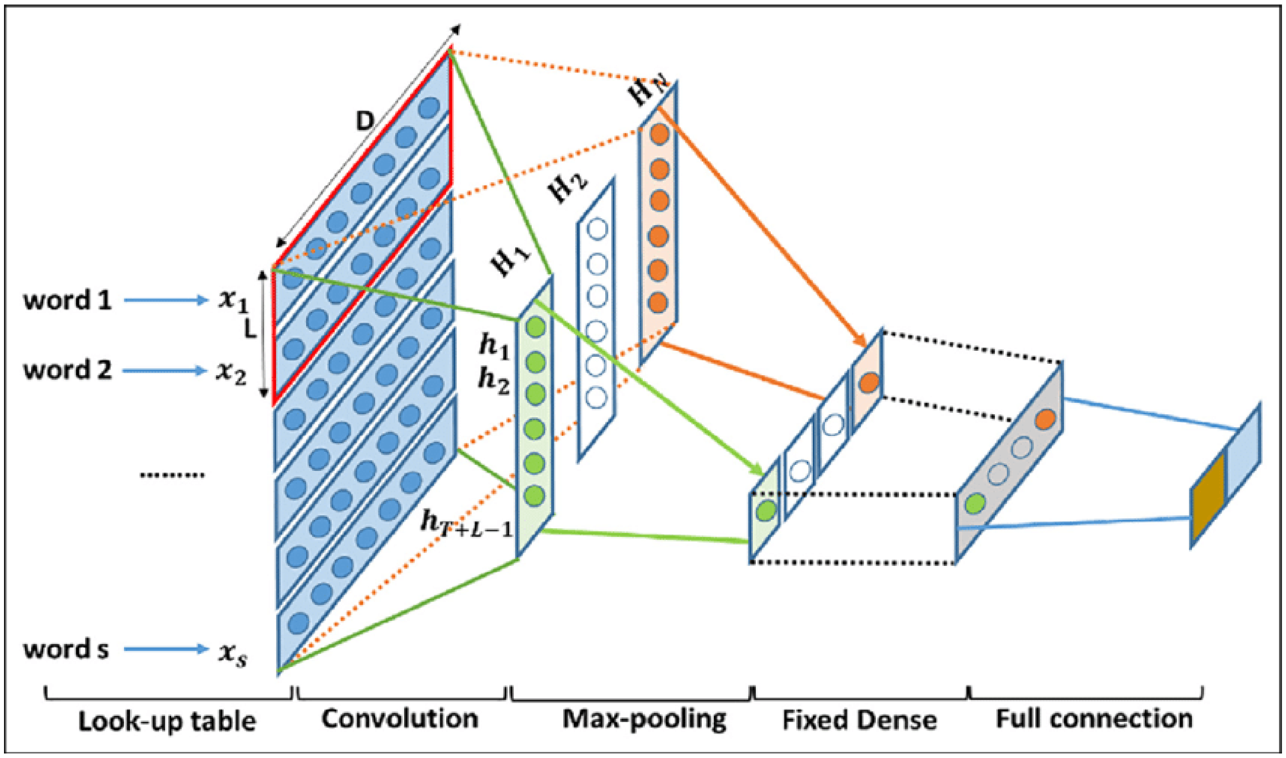


Figure 4. A CNN sequence to classify Image

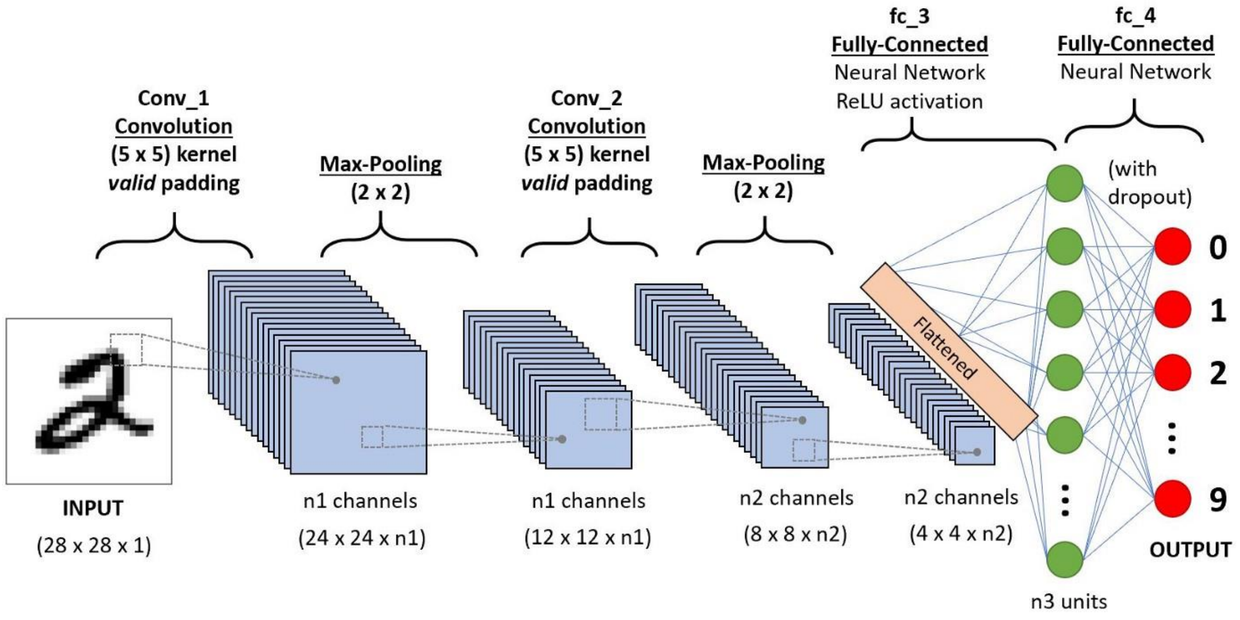


Figure 5. A CNN sequence to classify handwritten digits

Recurrent Neural Network (RNN):

A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit dynamic temporal behaviour for a time sequence.

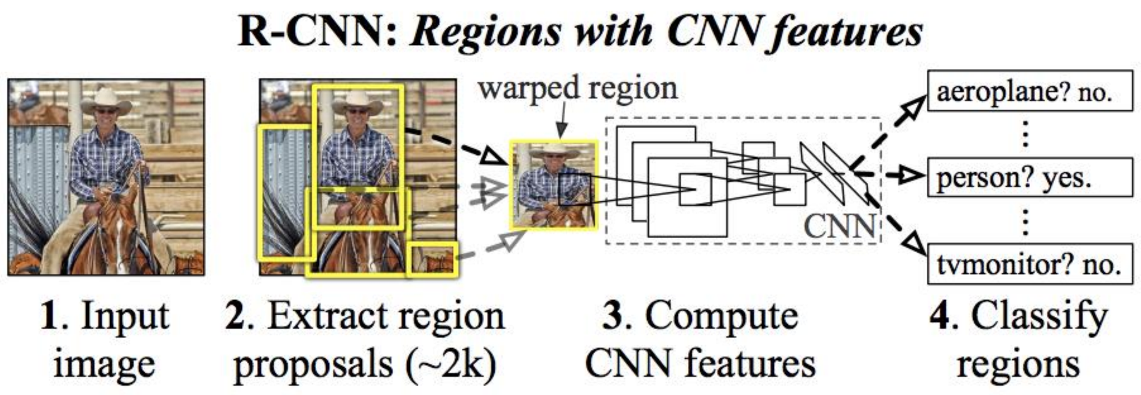


Figure 6. R-CNN

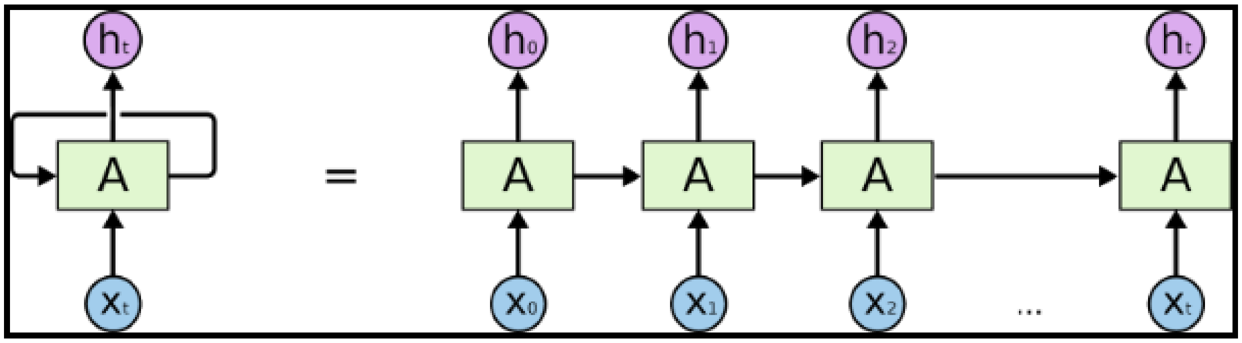


Figure 7 An unrolled recurrent neural network

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

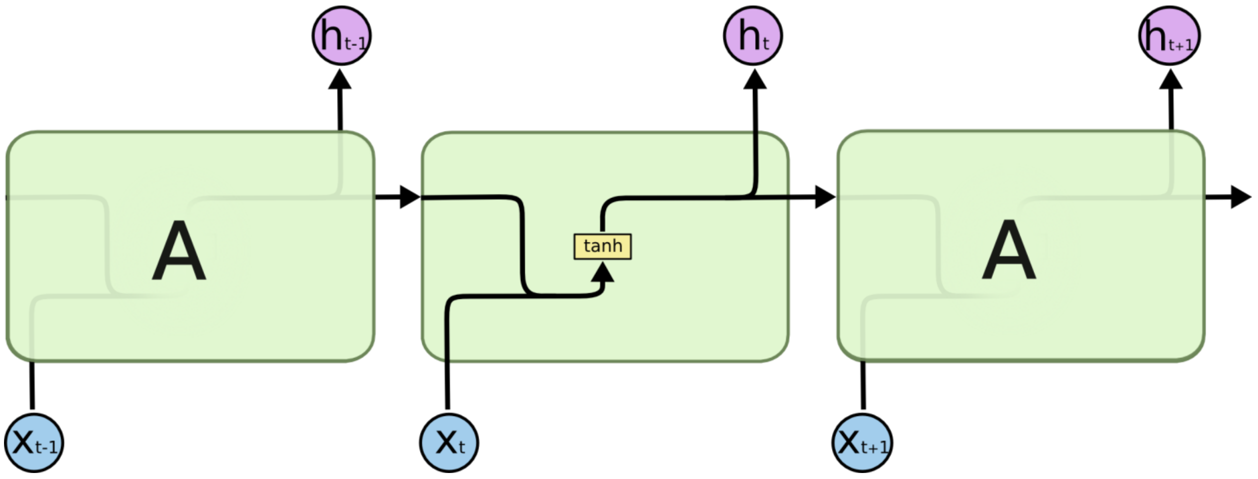


Figure 8 The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

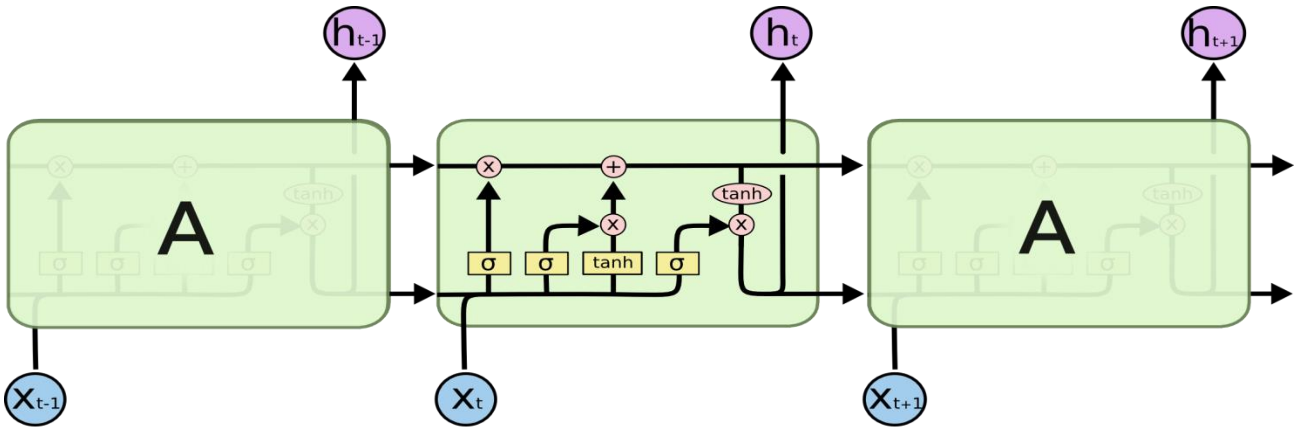
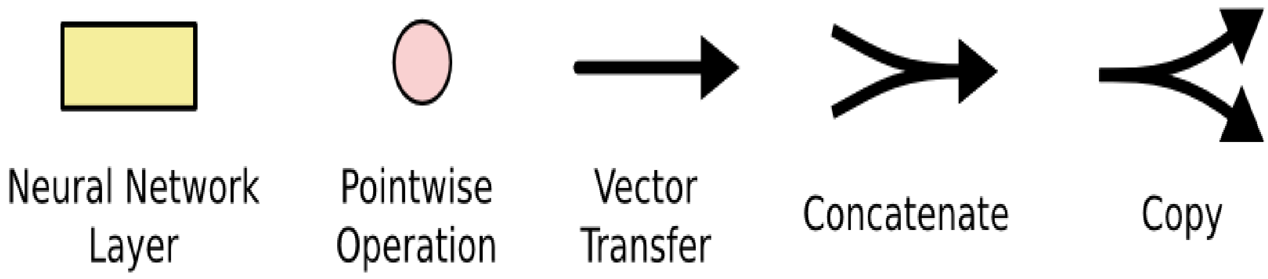


Figure 9.The repeating module in an LSTM contains four interacting layers



In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

Bidirectional long-short term Memory (Bidirectional LSTM) is the process of making any neural network o have the sequence information in both directions backwards (future to past) or forward (past to future).

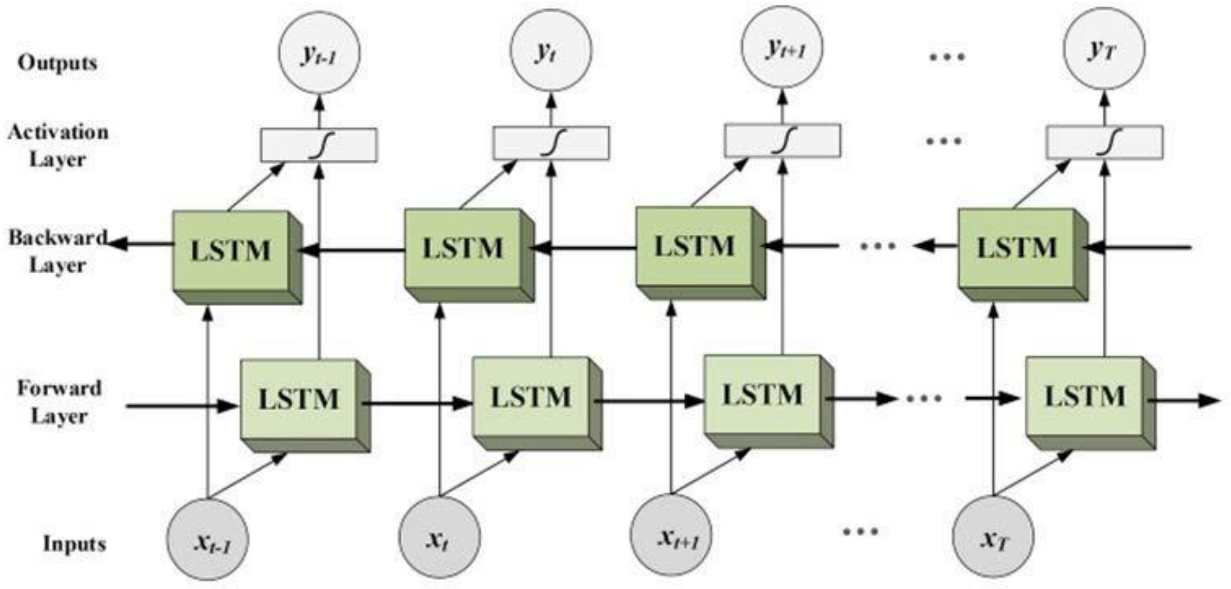


Figure 10. bi-LSTM

In the diagram, we can see the flow of information from backward and forward layers. BI-LSTM is usually employed where the sequence to sequence tasks are needed. This kind of network can be used in text classification, speech recognition and forecasting models.

CTC:

A Connectionist Temporal Classification Loss, or CTC Loss, is designed for tasks where we need alignment between sequences, but where that alignment is difficult - e.g. aligning each character to its location in an audio file. It calculates a loss between a continuous (unsegmented) time series and a target sequence.

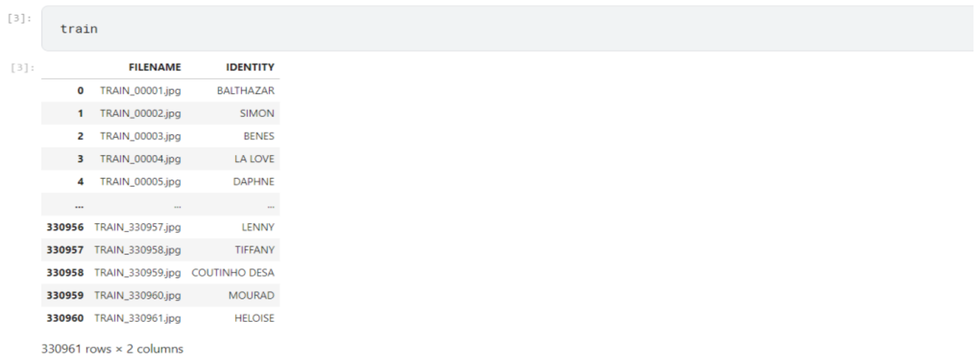
CHAPTER 7

TESTING AND RESULTS

Training & Testing

(We are using Kaggle handwriting-recognition Data Set for Testing and Training)

Exploratory Data Analysis (EDA)



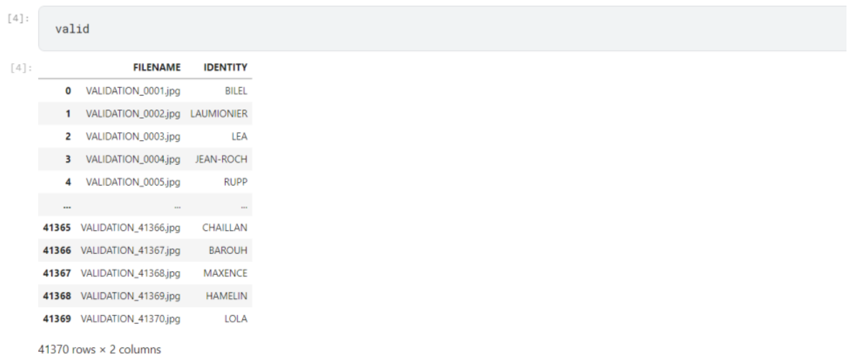
```
[3]: train
```

[3]:

	FILENAME	IDENTITY
0	TRAIN_00001.jpg	BALTHAZAR
1	TRAIN_00002.jpg	SIMON
2	TRAIN_00003.jpg	BENES
3	TRAIN_00004.jpg	LA LOVE
4	TRAIN_00005.jpg	DAPHNE
...	...	...
330956	TRAIN_330957.jpg	LENNY
330957	TRAIN_330958.jpg	TIFFANY
330958	TRAIN_330959.jpg	COUTINHO DESA
330959	TRAIN_330960.jpg	MOURAD
330960	TRAIN_330961.jpg	HELOISE

330961 rows × 2 columns

Figure 11. Training Dataset



```
[4]: valid
```

[4]:

	FILENAME	IDENTITY
0	VALIDATION_0001.jpg	BILEL
1	VALIDATION_0002.jpg	LAUMIONIER
2	VALIDATION_0003.jpg	LEA
3	VALIDATION_0004.jpg	JEAN-ROCH
4	VALIDATION_0005.jpg	RUPP
...	...	...
41365	VALIDATION_41366.jpg	CHAILLAN
41366	VALIDATION_41367.jpg	BAROUH
41367	VALIDATION_41368.jpg	MAXENCE
41368	VALIDATION_41369.jpg	HAMELIN
41369	VALIDATION_41370.jpg	LOLA

41370 rows × 2 columns

Figure 12. Validating Dataset

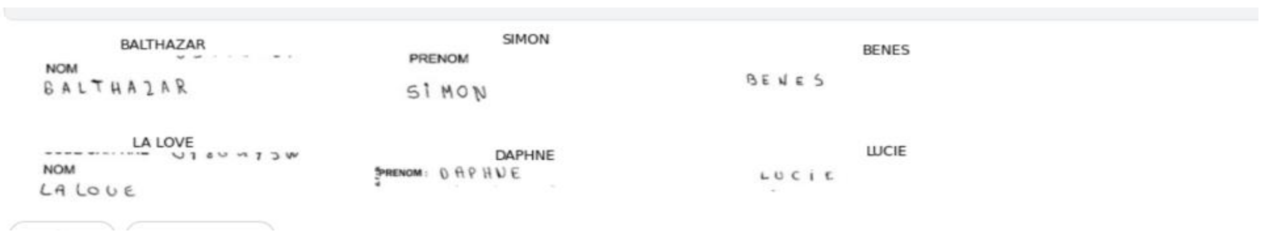
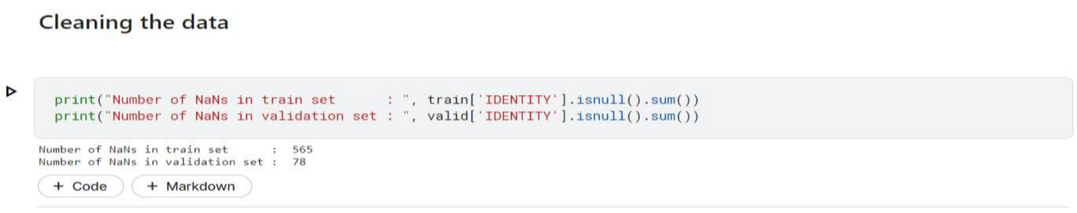


Figure 13. Analysis of Dataset

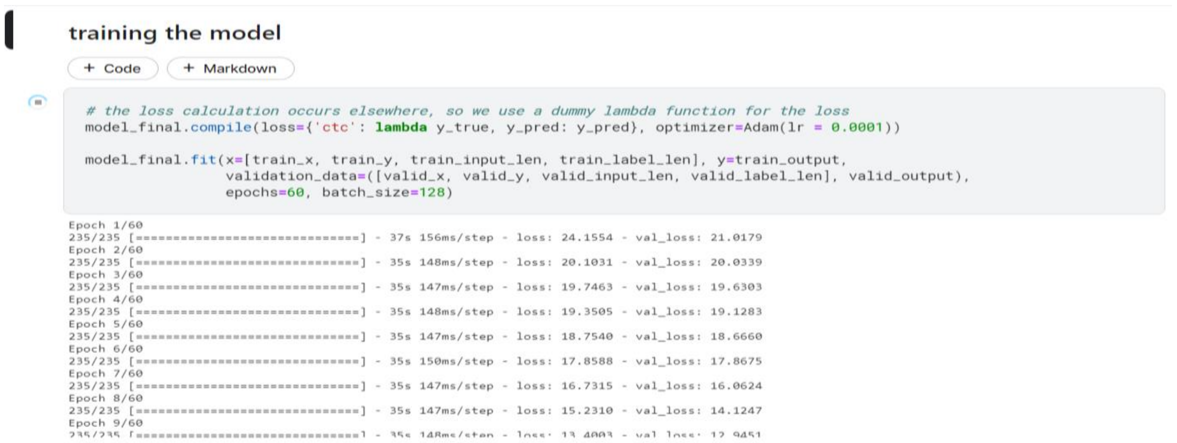


Cleaning the data

```
print("Number of NaNs in train set      : ", train['IDENTITY'].isnull().sum())
print("Number of NaNs in validation set : ", valid['IDENTITY'].isnull().sum())
```

```
Number of NaNs in train set      :  565
Number of NaNs in validation set :  78
```

Figure 14.Cleaning of Data



training the model

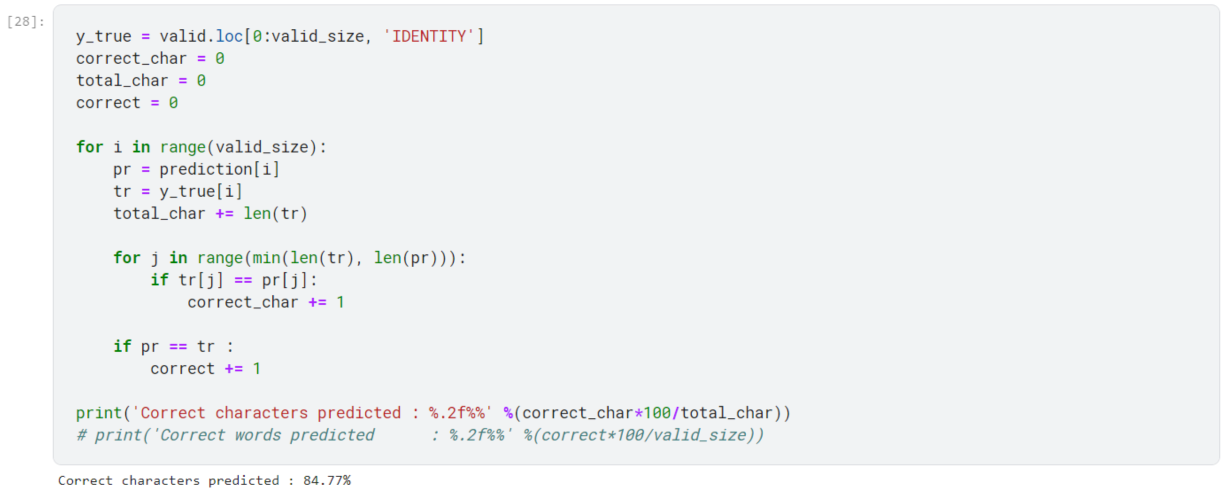
```
# the loss calculation occurs elsewhere, so we use a dummy lambda function for the loss
model_final.compile(loss={'ctc': lambda y_true, y_pred: y_pred}, optimizer=Adam(lr = 0.0001))

model_final.fit(x=[train_x, train_y, train_input_len, train_label_len], y=train_output,
                validation_data=([valid_x, valid_y, valid_input_len, valid_label_len], valid_output),
                epochs=60, batch_size=128)
```

```
Epoch 1/60
235/235 [==============================] - 37s 156ms/step - loss: 24.1554 - val_loss: 21.0179
Epoch 2/60
235/235 [==============================] - 35s 148ms/step - loss: 20.1031 - val_loss: 20.0339
Epoch 3/60
235/235 [==============================] - 35s 147ms/step - loss: 19.7463 - val_loss: 19.6303
Epoch 4/60
235/235 [==============================] - 35s 148ms/step - loss: 19.3505 - val_loss: 19.1283
Epoch 5/60
235/235 [==============================] - 35s 147ms/step - loss: 18.7540 - val_loss: 18.6660
Epoch 6/60
235/235 [==============================] - 35s 150ms/step - loss: 17.8588 - val_loss: 17.8675
Epoch 7/60
235/235 [==============================] - 35s 147ms/step - loss: 16.7315 - val_loss: 16.0624
Epoch 8/60
235/235 [==============================] - 35s 147ms/step - loss: 15.2310 - val_loss: 14.1247
Epoch 9/60
235/235 [==============================] - 35s 148ms/step - loss: 13.4003 - val_loss: 12.9451
```

Figure 15.Training of model

Accuracy Testing:



```
[28]:
y_true = valid.loc[0:valid_size, 'IDENTITY']
correct_char = 0
total_char = 0
correct = 0

for i in range(valid_size):
    pr = prediction[i]
    tr = y_true[i]
    total_char += len(tr)

    for j in range(min(len(tr), len(pr))):
        if tr[j] == pr[j]:
            correct_char += 1

    if pr == tr :
        correct += 1

print('Correct characters predicted : %.2f%%' %(correct_char*100/total_char))
# print('Correct words predicted      : %.2f%%' %(correct*100/valid_size))
```

```
Correct characters predicted : 84.77%
```

Figure 16. Accuracy Testing

Checking Performance on validation:

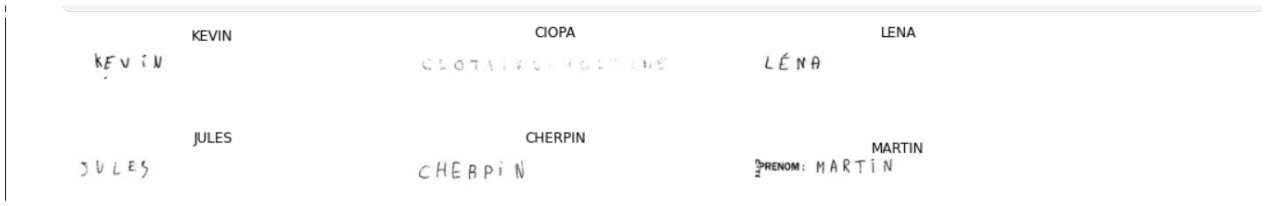
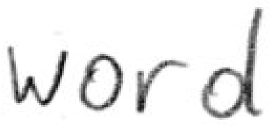


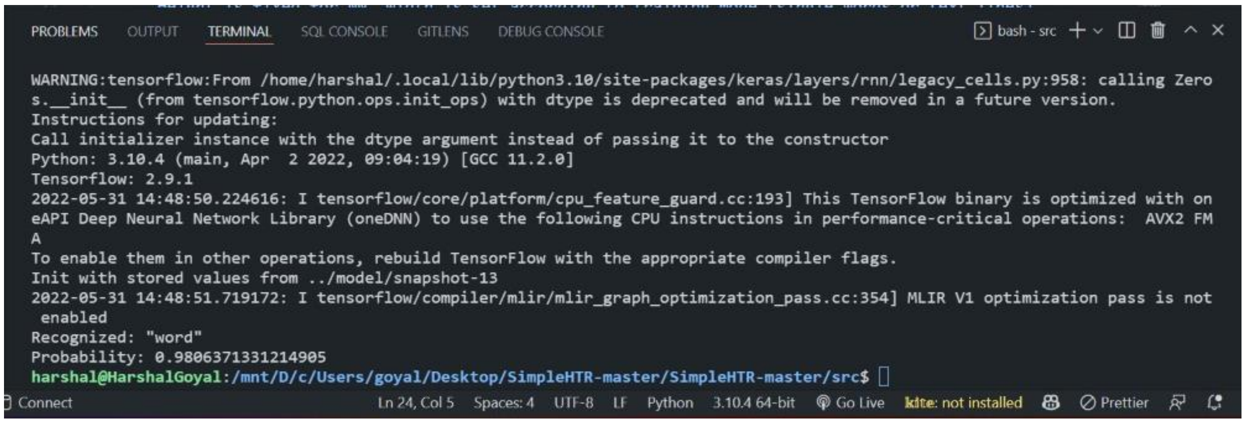
Figure 17. Output after validation

Implementation of Project:

Input Image1

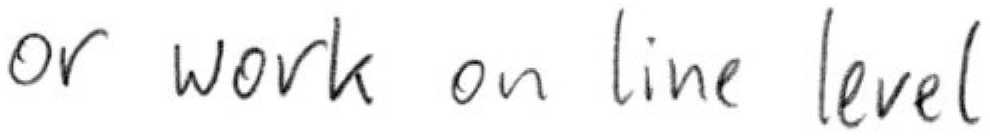


Output Snapshot of Input Image 2

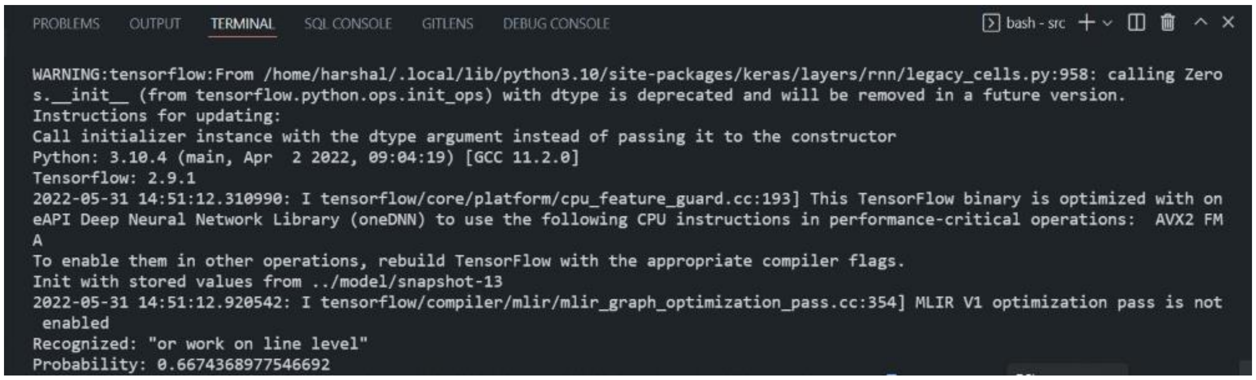


```
WARNING:tensorflow:From /home/harshal/.local/lib/python3.10/site-packages/keras/layers/rnn/legacy_cells.py:958: calling Zero
s.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Python: 3.10.4 (main, Apr  2 2022, 09:04:19) [GCC 11.2.0]
Tensorflow: 2.9.1
2022-05-31 14:48:50.224616: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with on
eAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX2 FM
A
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Init with stored values from ../model/snapshot-13
2022-05-31 14:48:51.719172: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:354] MLIR V1 optimization pass is not
 enabled
Recognized: "word"
Probability: 0.9806371331214905
harshal@HarshalGoyal:/mnt/D/c/Users/goyal/Desktop/SimpleHTR-master/SimpleHTR-master/src$
```

Input Image2



Output Snapshot Input Image1



```
WARNING:tensorflow:From /home/harshal/.local/lib/python3.10/site-packages/keras/layers/rnn/legacy_cells.py:958: calling Zero
s.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Python: 3.10.4 (main, Apr  2 2022, 09:04:19) [GCC 11.2.0]
Tensorflow: 2.9.1
2022-05-31 14:51:12.310990: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with on
eAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX2 FM
A
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Init with stored values from ../model/snapshot-13
2022-05-31 14:51:12.920542: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:354] MLIR V1 optimization pass is not
 enabled
Recognized: "or work on line level"
Probability: 0.6674368977546692
```

REFERENCES

[1] Mor, S. S., Solanki, S., Gupta, S., Dhingra, S., Jain, M., & Saxena, R. (2019). Handwritten text recognition: with deep learning and android. *International Journal of Engineering and Advanced Technology*, *8*(3S), 819-825.

[2] Manchala, S. Y., Kinthali, J., Kotha, K., Kumar, J. J. K. S., & Jayalaxmi, J. (2020). Handwritten text recognition using deep learning with Tensorflow. *International Journal of Engineering and Technical Research*, *9*(5).

[3]Balci, B., Saadati, D., & Shiferaw, D. (2017). Handwritten text recognition using deep learning. *CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University, Course Project Report, Spring*, 752-759.

[4]MOHAN KRISHNA, B., SWATHI, M., PRAKASH, H., & RAHUL, N. (2018). Handwritten Document Conversion Using Regional Convolutional Neural Network.

[5]Aqab, S., & Tariq, M. U. (2020). Handwriting recognition using artificial intelligence neural network and image processing. *International Journal of Advanced Computer and Application (IJACSA)*, *11*(7), 137-146.

[6]Hemanth, G. R., Jayasree, M., Venii, S. K., Akshaya, P., & Saranya, R. (2021). CNN-RNN Based Handwritten Text Recognition. *ICTACT Journal on Soft Computing*, *12*(1), 2457-2463.

[7]Obaid, A. M., El Bakry, H. M., Eldosuky, M. A., & Shehab, A. I. (2016). Handwritten text recognition system based on neural network. *Int. J. Adv. Res. Comput. Sci. Technol*, *4*(1), 72-77.

[8]Patel, D. K., Som, T., Yadav, S. K., & Singh, M. K. (2012). Handwritten character recognition using multiresolution technique and euclidean distance metric.

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

[10] L. Eikvil, "Optical character recognition," citeseer. ist. psu. edu/142042. html, 1993.

[11] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in Advances in neural information processing systems, 2009, pp. 545–552.

[12] 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

APPENDIX

CNN: Convolutional Neural Network

RNN: Recurrent Neural Network

BLSTM: Bidirectional Long Short-Term Memory

LSTM : Long Short Term Memory

CTC: Connectionist Temporal Classification

OCR: Optical Character Recognition.

SVM: Support Vector Machine