

Deep Learning Approach using CNN for Handwritten Character Recognition

Project report submitted in partial fulfillment of the requirement for the degree of

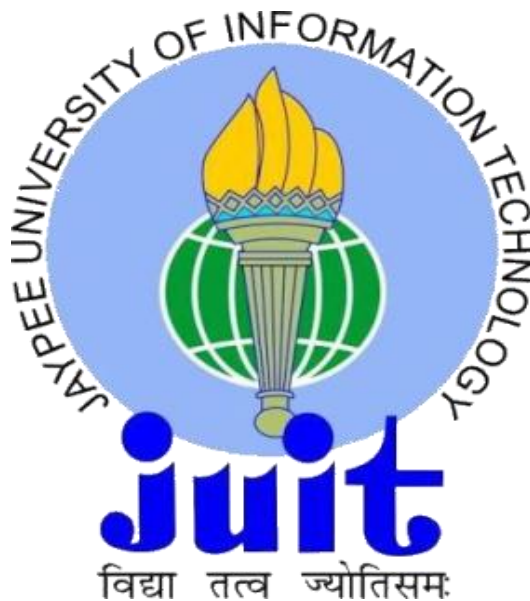
BACHELOR OF TECHNOLOGY IN ELECTRONICS AND COMMUNICATION ENGINEERING

By

Kamlesh Kumar (181016)

UNDER THE GUIDANCE OF

Dr. Pardeep Garg



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

May, 2022

TABLE OF CONTENTS

CAPTION	PAGE NO.
DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
CHAPTER-1: INTRODUCTION	1
CHAPTER-2: OBJECTIVE	2
CHAPTER-3: LITERATURE SURVEY	6
3.1 EARLY SCANNERS	6
3.2 TO THE DIGITAL AGE	6
3.3 MACHINE LEARNING	6
CHAPTER-4: PROPOSED WORK	9
4.1 WORKING PRINCIPLE	9
4.1.1 PRE-PROCESSING	11
4.1.2 SEGMENTATION	11
4.1.3 FEATURE EXTRACTION	11
4.1.4 CLASSIFICATION	11
4.2 TOOLS AND FRAMEWORKS	13
4.3 STEPS TO BUILD HANDWRITTEN CHARACTER RECOGNITION MODEL	14
4.3.1 IMPORTING LIBRARIES	14
4.3.2 READ THE DATA	14
4.3.3 SPLITTING THE DATA	14
4.3.4 REMODELING THE INFORMATION	14
4.3.5 FRAMING THE NUMBER OF ALPHABETS	15

4.3.6 REARRANGING THE DATA	16
4.3.7 RESHAPING DATA	17
4.3.8 IMPLEMENTING CNN MODEL	18
4.3.9 ACCURACIES	20
4.3.10 DOING SOME PREDICTION ON THE DATA	20
CHAPTER-5: RESULTS	22
5.1 OUTPUT	22
CHAPTER-6: CONCLUSION	29
REFERENCES	
APPENDIX	
PLAGIARISM REPORT	

DECLARATION

I hereby declare that the work reported in the B. Tech Project Report entitled “**Deep Learning Approach using CNN for Handwritten Character Recognition**” submitted at **Jaypee University of Information Technology, Wazirpur, India** is an authentic record of my work carried out under the supervision of Dr. Pardeep Garg. I have not submitted this work elsewhere for any other degree or diploma.

Kamlesh Kumar
181016

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Dr. Pardeep Garg
Date: 28/05/2022

Prof. Rajiv Kumar
Head of the Department

ACKNOWLEDGEMENT

I would like to express my uncommon thanks of appreciation to my instructor Dr. Pardeep Garg who offered me the brilliant chance to do this great project on the topic “Deep Learning Approach using CNN for Handwritten Character Recognition”, which additionally helped me for research the things that I am not familiar with and this project give time to me to think about new things, I am happy from the process which I’ve experienced. Also, I am happy to see that my friends and my parents for coordinating me and helped me in many things in restricted time period.

ABSTRACT

We humans rely so much over machines now a days that with the help of deep and machine learning algorithms, we can add sound to silent movies to classify objects in photographs. Similarly, handwritten text recognition is rich in terms of research and development with a number of outcome or possibilities that could be attained. Handwritten recognition is also known as handwritten text recognition (HTR). HTR having the ability interpret and receive handwritten input from sources like touch screens, images, paper document, etc. In this project, I have performed handwritten character recognition or we can say handwritten text recognition with the help of A-Z Handwritten data from Kaggle using Convolution Neural Network (CNN) models. The main objective of this project is to build a model to predict the character with input given by the user and to see the model accuracy with their execution time to find the best possible model for handwritten text recognition. Because it is still a great challenge to get the precise recognition rate of having complex shaped texts. There is a lot of recent progress in Convolution Neural Network (CNN) and it is directly proportional to Handwritten Character Recognition by studying biased attribute from immense measures of raw information. This program will be used to find documents in different formats. The development of handwriting is very complex, with different types of characters like digits, number, compound text & texts combining different languages.

CHAPTER I

INTRODUCTION

Handwritten recognition empowers to interpret various sorts of reports into analyzable, editable and accessible information. An extreme point of Handwritten Character Recognition is to copy peoples perusing capacities so that the model can peruse, alter and collaborate with text in brief time frame. Recognizable proof of Handwritten Character Recognition has drawn extraordinary consideration of various scientists over 50 years, and numerous incredible accomplishments have been made in this field [1]. In current situations, we have so much specialized advancements like 3D finger that can fill the left space or catch any movement with the help of computer vision. Be that as it may, in the previous years, a critical advancement is made on HCR execution, yet at the same time now HCR is a moving assignment because of the incredible variety of style in handwriting, the presence of numerous comparative texts& huge no. of texts classifications [2].

Exploiting the new remarkable development in the size of clarified information and the fast expansions in the capacities of the illustrations handling components, the concentrate on Convolution Neural Network has rapidly emerged and acquired cutting edge execution on various assignments, e.g., picture order, text identification, present assessment, object following, activity recognition, visual saliency detection, scene stamping, and NLP. Despite the fact that there are numerous variations of CNN structures, their essential components are practically the same. Numerous analysts have done research in HCR field yet 100% exactness can't be accomplished.

My point is to foster an effective Handwritten Character Recognition model utilizing Convolution Neural Network. To test the HCR system A-Z dataset from Kaggle has been utilized. To survey the exhibition of CNN calculation, I tried different things with the A-Z dataset and tracked down the precision of manually written characters. The pictures are parted into preparing and test sets. Preparing is completed with different pictures backthen, at that point, testing is led to track down the accuracy of the CNN.

CHAPTER 2

OBJECTIVE

Pattern recognition has one domain that is handwritten recognition. The purpose of pattern recognition is to classify or separate the data or object into one of the categories or categories. Handwriting recognition is defined as the task of translating the language represented by its geographical location of the image tags in its symbolic representation. Each text has a set of thumbnails, also known as letters or letters, with a basic shape.

The goal of handwriting is to identify the input characters or the image correctly and to analyze it in most automated process systems. This program will be used to find documents in different formats. The development of handwriting is very complex, with different types of handwritten characters such as digit, number, compound text, symbols, and texts combining English and other languages.

Automatic recognition of handwritten text can be very useful in many applications where it is necessary to process large amounts of handwritten data, such as visual address and postal codes in envelopes, analysis of bank check rates, document analysis, and signature verification. Therefore, a computer is needed to read a document or data to make it easier to process documents. Machine learning algorithms that feed computer data into AI systems, which use mathematical techniques to learn AI systems. By mechanical learning, AI systems are gradually improving in performance, unless they are specifically programmed to do so.

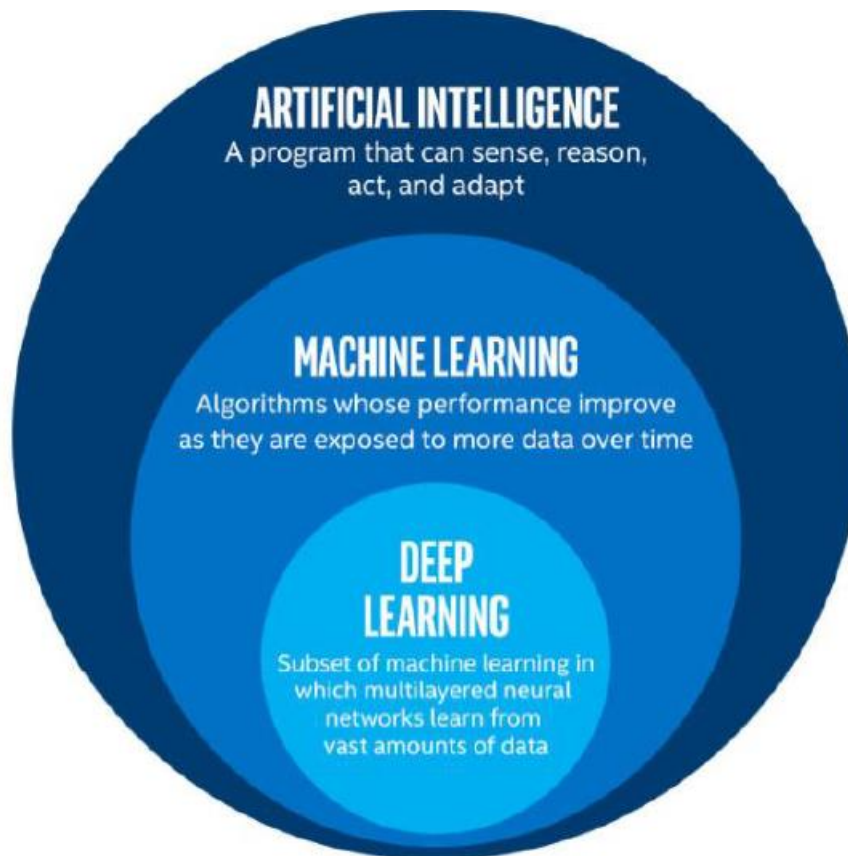


Fig 2.1 - AI, ML, and DL in one Diagram

Artificial intelligence is whole thing and in that Deep Learning and Machine Learning is a part of AI. In-depth these algorithm goes as same conclusion where human go but with the structural manner in a logical way. And to conclude the result or finding the output these in-depth learning algorithm uses structured network that is neural network.

Like the structure of our human brain, neural network is also based on that structure. We use to classify different things and patterns with separate type of information likewise the work done by the neural network is also the same.

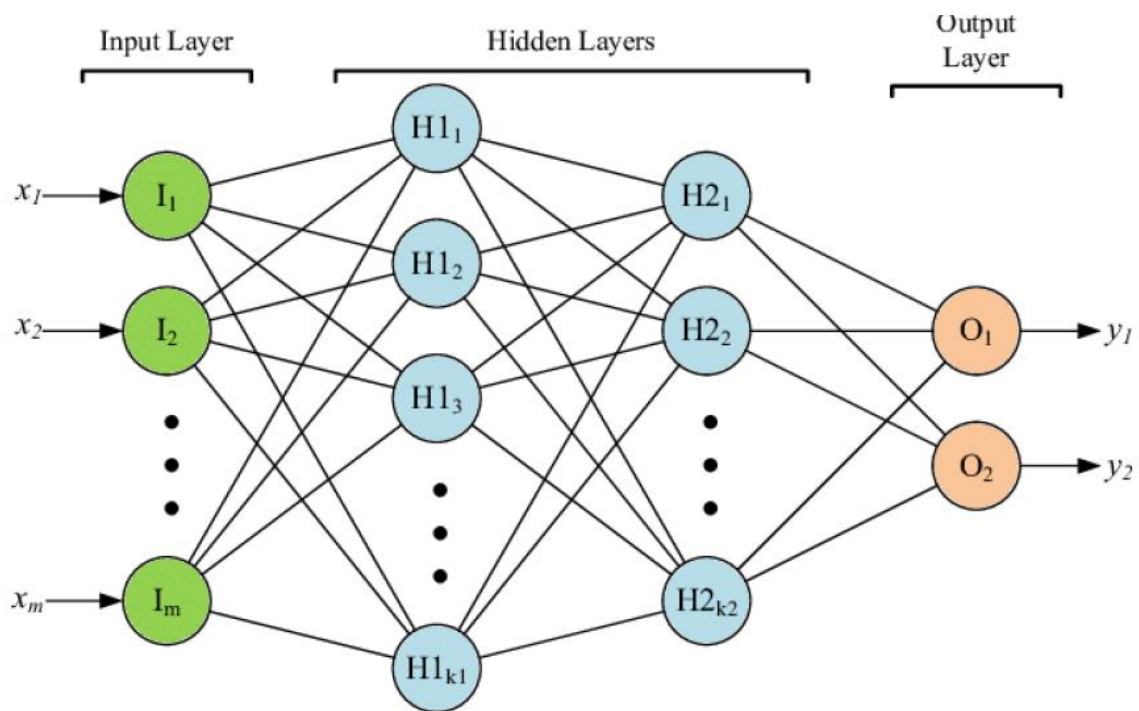


Fig 2.2 – Deep Learning NN Architecture

NLP is directed or we can say related with computer science department and in particular, the field of AI which deals to provide simple text to read from computers in the similar way humans can. Natural language processing incorporates modeling based on human language mathematics rules, machine, and in-depth models. Combined, these technologies allow computers to 'understand' its complete, complete meaning and purpose of the speaker or author and his or her mood. There are many opportunities for us to interact with NLP through GPS voice systems, digital assistants, speech-to-text call software, customer service chatbots, and other consumer resources.

Based on the image resolution, it will see $h \times w \times d$ (h = Length, w = Width, d = Size). Eg. Picture of $6 \times 6 \times 3$ identical members of the RGB matrix (3 refers to RGB values) and $4 \times 4 \times 1$ members of the same gray matrix members.

The following common layers of CNN:

- 1) **Convolution-** CNN's main building block is a layer of convolution that it combines two data or row or column data. We convolute any input signal or data using convolution filter to make a feature map. The first step in CNN algorithm is to go through the convolution.
- 2) **Stride-** How many times shift is occurring over the inserted picture is known as stride. Like if stride is 4 then moving the filters to 4 pixels at a time.
- 3) **Non-Linearity** - The most commonly used Activation function states that ReLU represents the Rectified Linear Unit to operate offline.
- 4) **Pooling layer** - After convolution operation we often put it together to reduce the size. Combining the layers down with a sample map of each element independently, reduces the length and width, keeping the depth unchanged. In contrast to convolution work, merging has no limits.

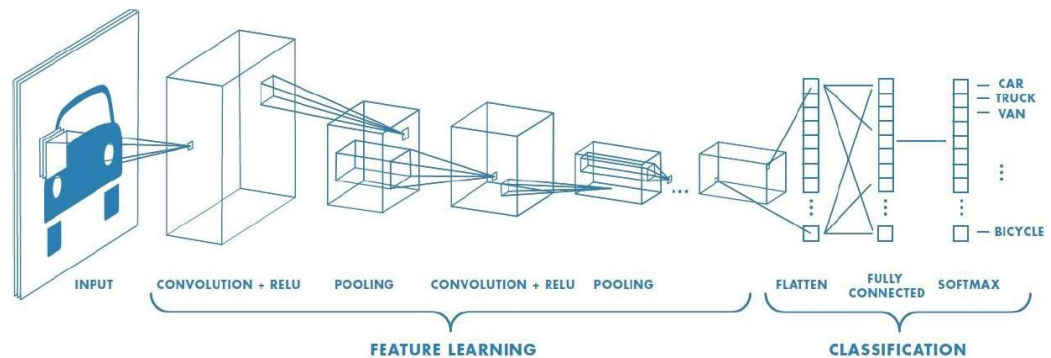


Fig 2.3 - CNN Architecture

CHAPTER 3

LITERATURE SURVEY

3.1 Early Scanners

Jacob Rabinows has done research very early in his days. His research was to scan a single-spaced font like one character at a moment using scanning hardware designed rationale to perceive that. And this is first research in handwritten character classification and his project is digit classification for postal mail [3]. Later Allum *et. al* worked on this which has more verities in how the text was composed just as encoding the data onto a standardized tag that was printed straightforwardly on the letter by making a sophisticated scanner [4].

3.2 To the digital age

The primary conspicuous section of Optical Character Recognition programming was created by Ray Kurzweil in 1974 as the product considered acknowledgment for any text style [5]. This product utilized a more evolved utilization of the network technique (design coordinating). Basically, this would contrast them to figure out what character it most firmly coordinated with. The drawback was this product was touchy to varieties in measuring and the differentiations between every individual's method of composing. For each character, programming would search for highlights like predictions, histograms, drafting, and mathematical minutes [6].

3.3 Machine Learning

Lecun *et. al* focused on utilizing inclination-based learning strategies utilizing multi-module AI models, an antecedent to a portion of the underlying start to finish present day profound learning models [7].

The following significant update in creating high OCR correctness's was the utilization of a Hidden Markov Model for the errand of OCR. This methodology utilizes letters as a state, which then, at that point, takes into consideration the setting of the person to be represented while deciding the following secret variable [8]. This led to higher precision contrasted with both component

extraction methods, and the Naive Bayes approach [9]. The fundamental downside was as yet the manual extraction highlights, which needs earlier information on the language and was not especially powerful to the variety and complexity of handwriting.

Ng et. al applied CNNs to the issue of taking text found in the wild (signs, composed, and so forth) and recognized text inside the picture by utilizing a sliding window. The sliding window gets across the picture to observe an expected example of a person being available. A CNN with two convolutional layers, two normal pooling layers, and a completely associated layer was utilized to characterize each character [10].

Yan utilizes the Faster RCNN model [11] to recognize individual characters inside a word and for characterization. This uses a sliding window across the picture to initially decide if an item exists inside the limits. That limited picture is then arranged to its relating character. Yan likewise carries out alter distance which takes into consideration making changes to the arranged word to decide whether one more characterized word is bound to be right [12].

We decided to use the MNIST database [26], with a few successes using this database. Even before using Deep Reading, the manuscript recognition was made possible, yet your accuracy was really low, or they had a small database as suggested by Line Eikvil [27]. In this paper, the use of OCR is discussed such as Speech Recognition, Radio Frequency, Visual Systems, Magnetic Stripes, Bar Code and Visual Symbol Readings. A popular machine learning function is to separate the MNIST database, which is a numerical database.

Enhanced English Handwriting Recognition Using Teddy Surya's Deep Neural Network with Ahmad Fakhur using the Deep NN model with two layers and one-layer SoftMax in the MNIST database. Their accuracy using Deep Neural Network was good comparatively to previously proposed net with pattern & feed-forward ANN. [31] Model obtained more than 87 percent accuracy using Convolutional Neural Networks from the Camera library [33].

The integrated model is built using a combination of several CNN models. Recognition tests were performed for MNIST digits, and 99.73% accuracy was reported [34]. Later, the "7-net committee" was expanded to "35-net committee" tests, and improved recognition accuracy was reported as 99.77% in the same MNIST database. An exceptional 99.81% recognition accuracy was reported by Niu and Suen for combining support vector machine capabilities to reduce structural risk and the ability of CNN model to extract in-depth features of MNIST digital recognition testing [26]. However, through careful research, it has been noted that high image accuracy of MNIST data sets is achieved. using only integration methods. Integration methods help to improve the accuracy of the sections but with the cost of complex testing tests and the growing computer costs of real application. The training set consists of 697932 pictures and the test set contains 116323 characters for upper- and lower-case letters and numbers from 0-9 designed for their corresponding classes. The Y train and the Y test are both lists containing numbers from 0 to 61 as there are 10 numbers from 0-9 as well as 26 uppercase letters and lowercase letters each comprising up to 62. [26]

CHAPTER 4

PROPOSED WORK

This project is basically handwritten character recognition, means recognizing characters from A to Z. Kaggle dataset which contains images of alphabets has been used to train a model that can help in achieving the target and recognize characters with good accuracy. The images in this dataset contain 37245 images of alphabets. This dataset is in the form of CSV file. [35]

4.1 Working Principle:

Handwriting Recognition System (HTR) systems contain handwritten text in the form of scanned images. we will build a Neural Network (NN) trained in word pictures from the IAM database.

A block diagram of the proposed hand-drawn character program. The proposed detection method relies on a convolutional neural network model (CNN) with an exit layer mapped. Our proposed model will split the input provided in 10 classes using CNN while dividing the number.

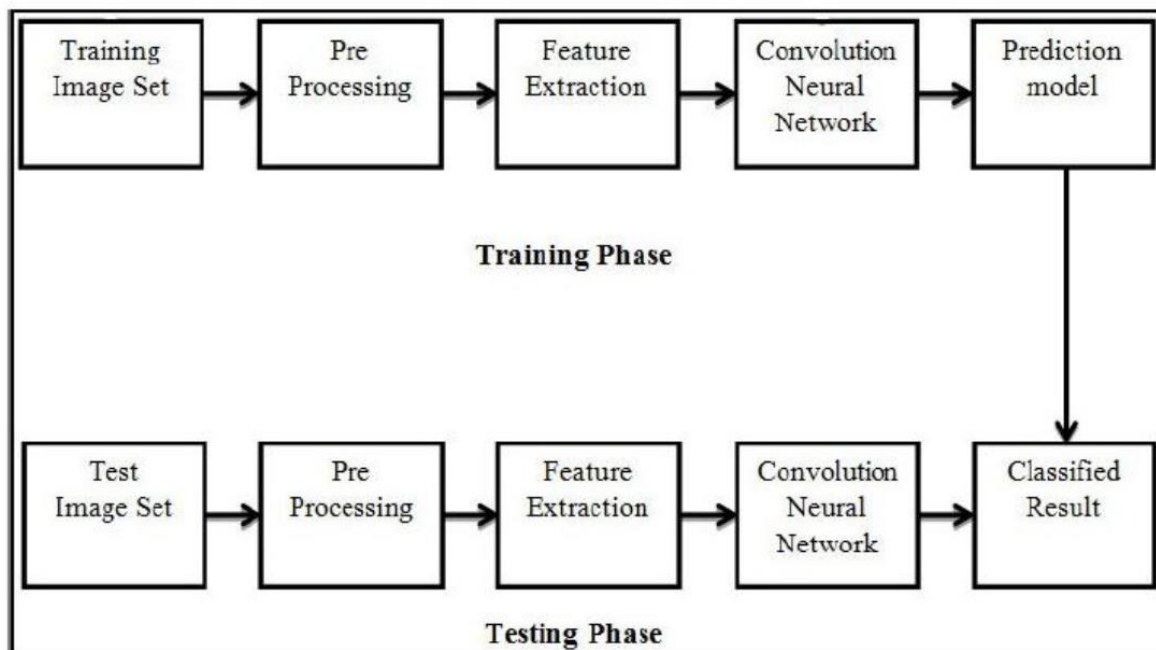


Fig 4.1 HCR Block diagram

The same process model is used repeatedly in the development of many applications and thus, has many advantages. Another method that can be used as a process model is to describe how things should be done as opposed to the process itself actually occur.

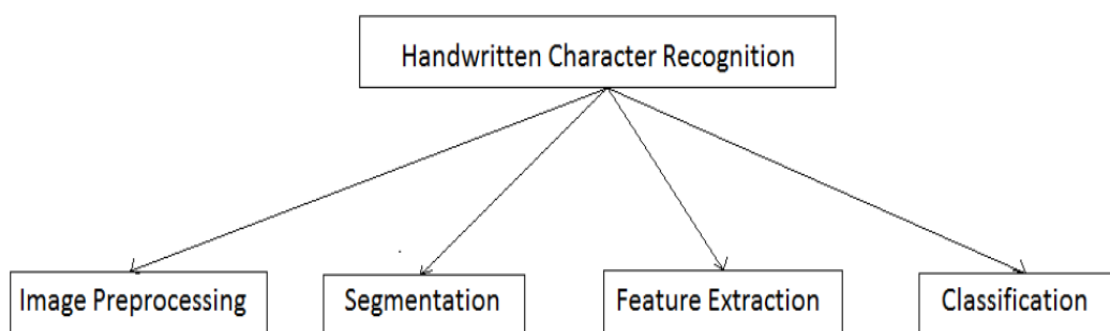


Fig 4.2 Breakdown model

4.1.1 Image pre-processing

Pre-processing is the essential period of character acknowledgment and it's vital for great acknowledgment rate. The principle objective of pre-handling steps is to standardize strokes and eliminate varieties that would some way or another confuse acknowledgment and decrease the acknowledgment rate. Pre- processing incorporates five normal advances, specifically, size standardization and focusing, introducing missing focuses, smoothing, incline amendment and resampling of focuses.

4.1.2 Segmentation

Division is finished by detachment of the singular characters of a picture. By and large, record is handled progressively. At first level lines are divided utilizing line histogram. From each line, words are extricated utilizing segment histogram lastly characters are separated.

4.1.3 Feature extraction

The main aim of feature extraction phase is to extract that pattern which is most relevant for classification. Feature extraction techniques like Principle Component Analysis (PCA), Gradient based features, Histogram, etc might be applied to extract the features of individual characters. These features are used to train the system.

4.1.4 Classification

When input image is presented to HCR system, its features are extracted and given as an input to the trained classifier like artificial neural network or support vector machine. Classifiers compare the input feature with stored pattern and find out the best matching class for input.

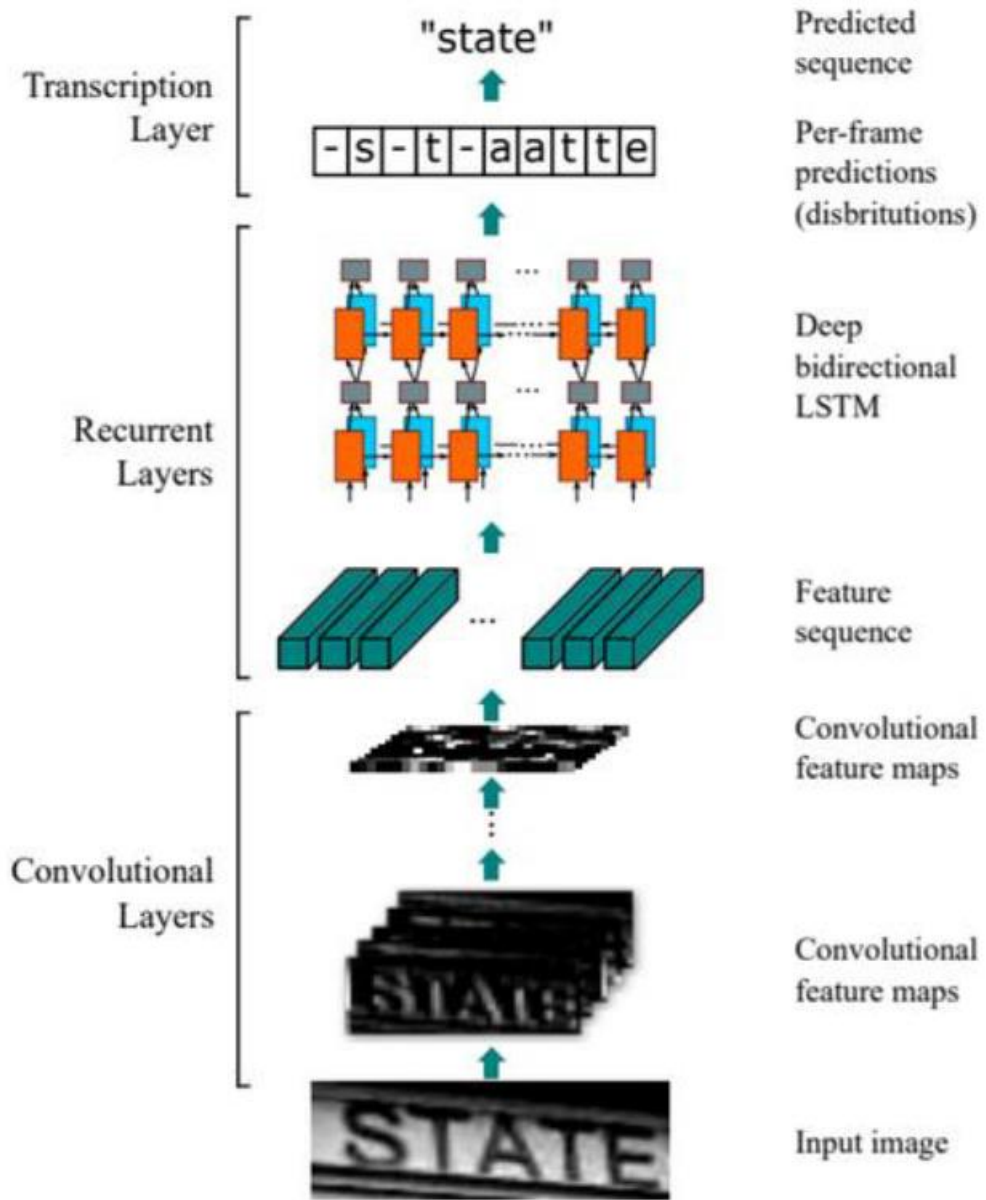


Fig 4.3 Proposed Model Architecture

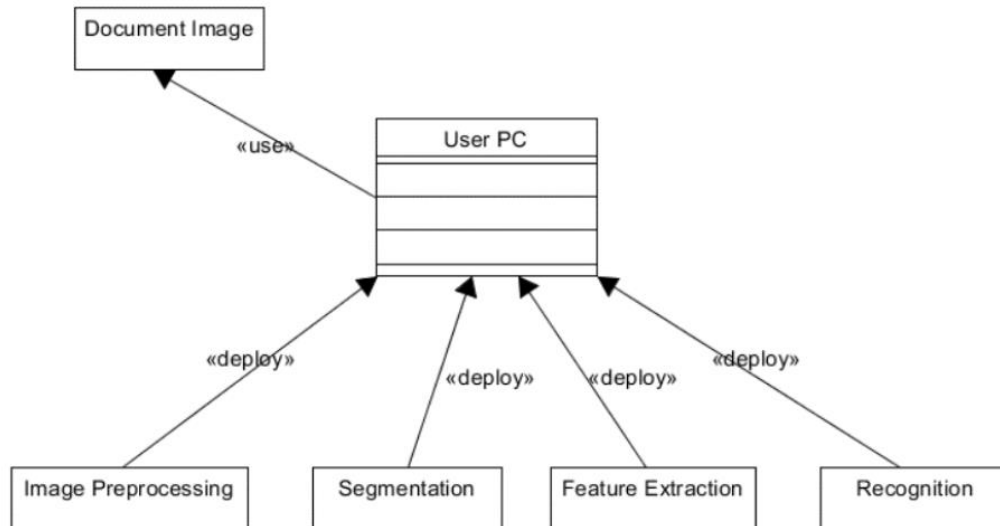


Fig 4.4 Deployment Diagram

For the appropriate database selected by us, the data set will be uploaded and a small portion of it will be used, after splitting it into a training and testing set. First, we aim to develop a compact architectural model for Convolution neural network, Recurrent Neural Network and other desirable layers, using Python DL libraries. Then we look at how we first do the pre-processing tasks that need to be done in the database, to deliver the data and then go to the Train the Model. After will train more models by changing their parameters slightly so that we have more models to test.

4.2 Tools and frameworks

This project requires some tools and frameworks:

- Python
- Jupyter as IDE
- NumPy
- CV2
- TensorFlow
- Keras
- Matplotlib
- Pandas

4.3 Steps to build handwritten character recognition Model:

(All the codes written for this project is available in appendix)

4.3.1 Importing Libraries

In the beginning of every python Machine learning or deep learning projects, we have to import the necessary in-built libraries.

Import mnist from Keras.datasets: Keras.datasets is a module that provides few vectorized datasets in NumPy format. It can be used for debugging a model. Mnist dataset stands for the modified national institute of standards and technology dataset that contains small number of datasets.

4.3.2 Read the data:

Pandas work is to read the datasets using `pd.read_csv ()` with float32 type. So that it is accessible to all the codes down below without giving any error.

4.3.3 Splitting the data:

Splitting this into X and Y i.e. The data (images) and the predict label. The 0-column having the corresponding labels and dropping 0 from axis 1 i.e., column from the dataset and use it in the Y to utilize the names separately.

4.3.4 Remodeling the information:

Remodeling the data and for that splitting it into the csv file. Also splitting the rest dataset into testing and preparing dataset. Likewise, Reshaping the test and train images information so they can be shown as a picture, as at first in the CSV record, they were available as 784 segments of pixel information.

```
Train data shape: (297960, 28, 28)
Test data shape: (74490, 28, 28)
```

Fig 4.5 – Output of Training and Testing data values

Every one of the names are available through drifting spike values, which requires the transformation to numeric qualities and that were making a word reference like a dictionary to plan the number values with the characters.

4.3.5 Framing the number of alphabets

Right off the bat, where transforming the marks into numeric values and add it into the count list as per the name. Just portraying here, the distribution of the letter sets.

See the horizontal bar plot between number of element and alphabets which was taken from output in fig 4.6

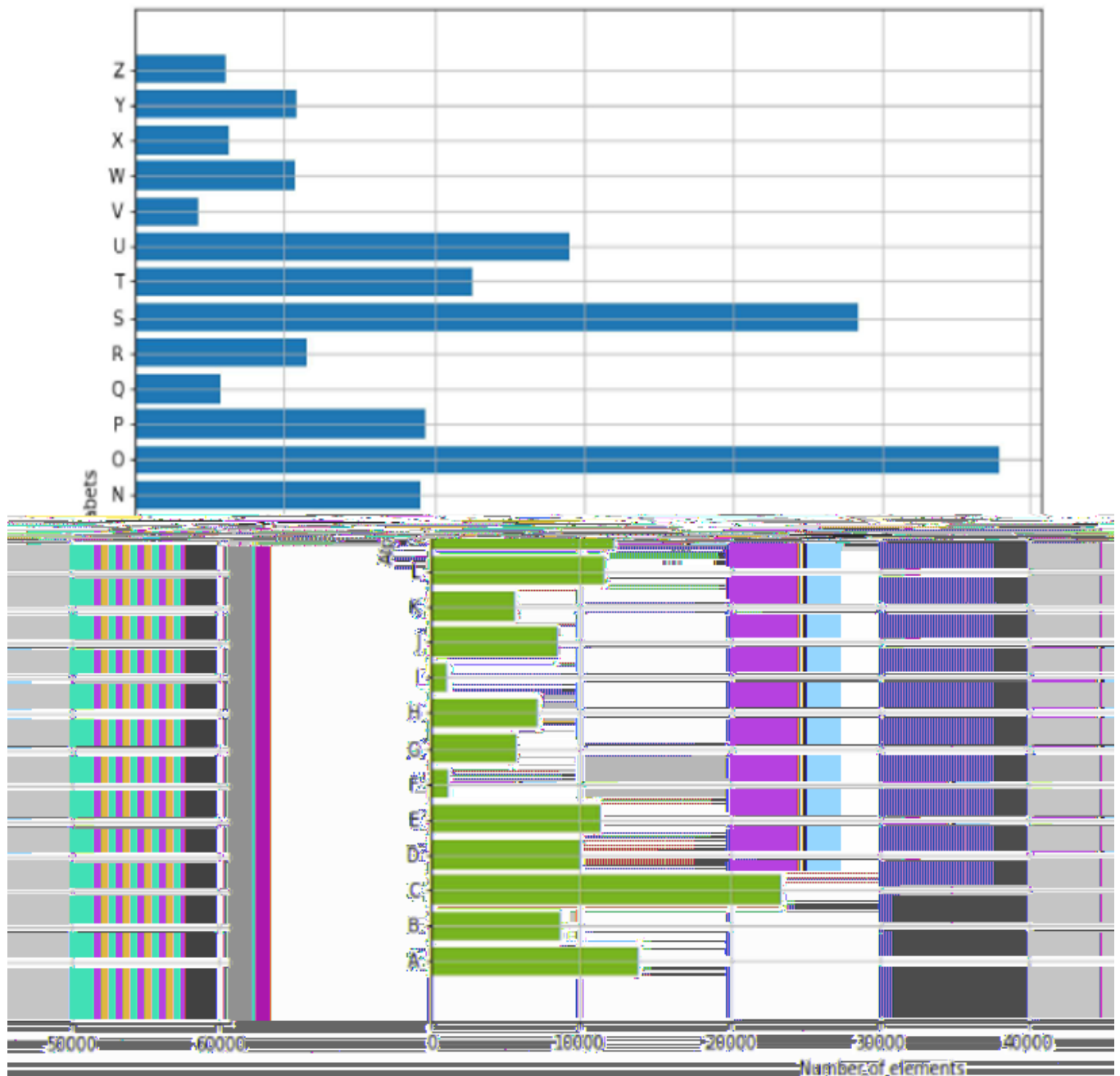


Fig 4.6 – Horizontal bar plot b/w number of elements and alphabets

4.3.6 Rearranging the data

In order to get the best accuracy while getting the different result every time in prediction by the model, rearrange or shuffle the data using shuffle () function in python. Doing some shuffling on the training datasets that is filled with images. In result, don't want all the images in training datasets display on the screen, that's why created 9 plots like a matrix in 3 x 3 shape in Fig 4.7. And all the images are displayed on the screen as output is a greyscale image.

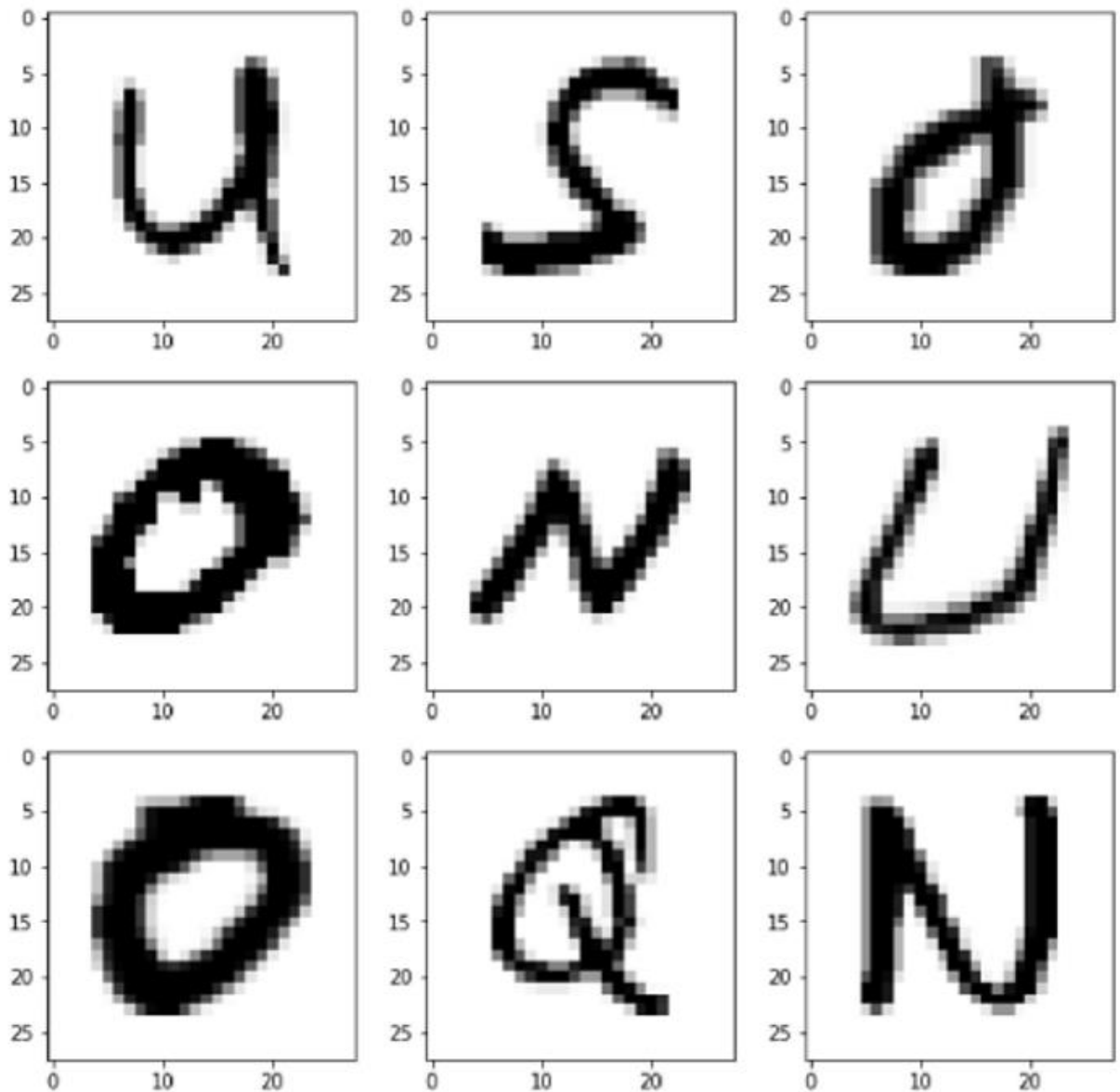


Fig 4.7 – Matrix shaped image output

4.3.7 Reshaping Data

Now reshaping the testing and training data again without having any error or mis classification in the model.

`New shape of train data: (297960, 28, 28, 1)`

New shape of test data is (74490, 28, 28, 1). So, from now onwards, my HCR model using CNN algorithm taking the labels input and generates possible outputs as a vector.

The new shapes for:

Train labels: (297960, 26)

Test labels: (74490, 26)

New shape of train labels: (297960, 26)

New shape of test labels: (74490, 26)

4.3.8 Implementing CNN model:

Now here comes the main function of this model, using this CNN algorithm, the model that can filter the input and give us back the polished output.

CNN :

CNN stands for convolution neural networks that are used to extract the features of the images using several layers of filters.

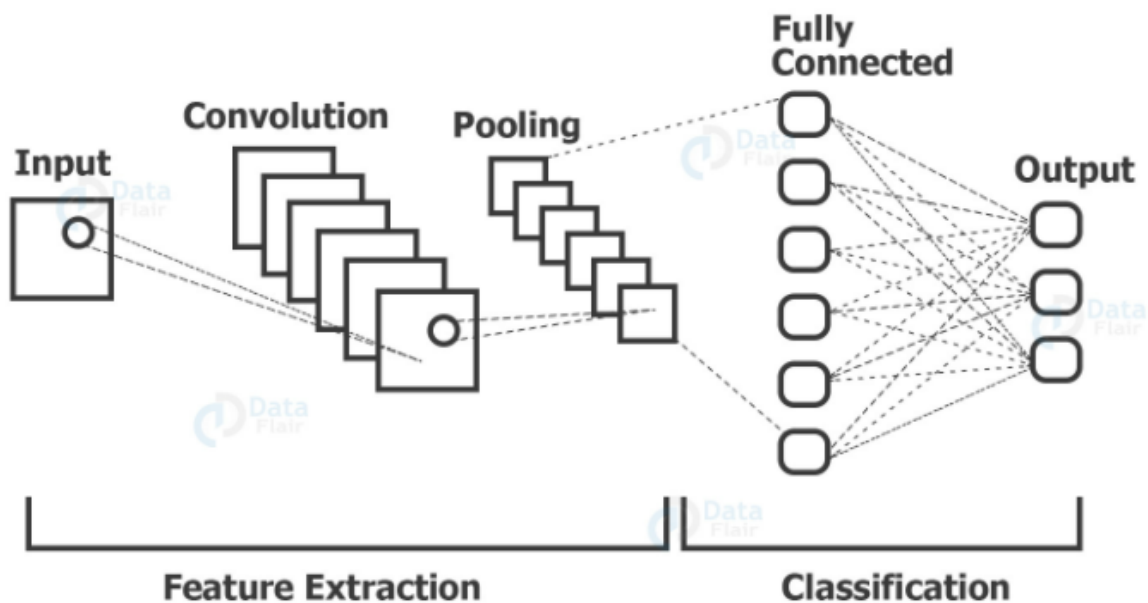


Fig 4.8 – Working diagram of CNN

The CNN model built for training the models using training dataset.

Arranging the model, where characterized the improving capacity and the misfortune capacity to be utilized for fitting. This data is exceptionally enormous so preparing for just a solitary age, in any case, as required we can even train it for a very long time which is suggested for character acknowledgment for better precision.

```
WARNING:tensorflow:From C:\Users\Bhupender Yadav\anaconda3\lib\site-packages
\keras\backend\tensorflow_backend.py:4070: The name tf.nn.max_pool is depreca
ted. Please use tf.nn.max_pool2d instead.
```

```
WARNING:tensorflow:From C:\Users\Bhupender Yadav\anaconda3\lib\site-packages
\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is dep
recated. Please use tf.compat.v1.global_variables instead.
```

```
Train on 297960 samples, validate on 74490 samples
```

```
Epoch 1/1
```

```
297960/297960 [=====] - 413s 1ms/step - loss: 0.1679
```

```
- accuracy: 0.9547 - val_loss: 0.1029 - val_accuracy: 0.9720
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_2 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_3 (Conv2D)	(None, 4, 4, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 128)	0

flatten_1 (Flatten)	(None, 512)	0
dense_1 (Dense)	(None, 64)	32832
dense_2 (Dense)	(None, 128)	8320
dense_3 (Dense)	(None, 26)	3354
=====		
Total params: 137,178		
Trainable params: 137,178		
Non-trainable params: 0		

Fig 4.9 – Output of Parameters or filters that the model has gone through

4.3.9 Accuracies

The validation accuracy is: 0.9719962477684021

The training accuracy is: 0.95470536

The validation loss is: 0.10290976331905201

The training loss is: 0.16793842845979098

```
The validation accuracy is : [0.9719962477684021]
The training accuracy is : [0.95470536]
The validation loss is : [0.10290976331905201]
The training loss is : [0.16793843845979098]
```

4.3.10 Doing Some Predictions on Test Data

To predict model on the test data after training the model with the training data and after that having a good accuracy of training as well as in validation too.

(74490, 28, 28, 1)

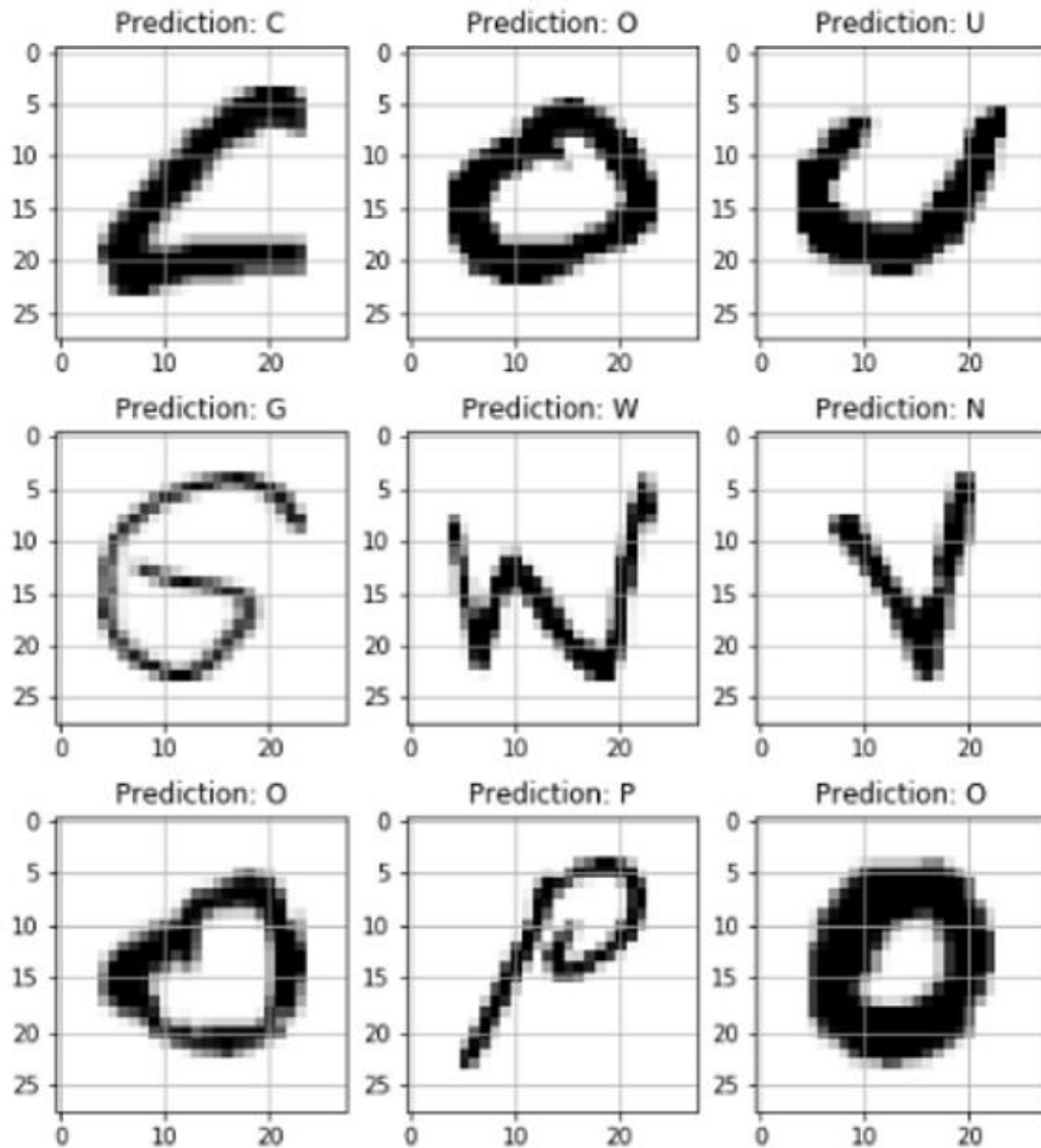


Fig 4.10 – Output of Prediction on test data

As you can see here in Fig 4.10, the Outputs of self generated by the model and looking at it, it looks like having almost 95% of accuracy in the outcome.

CHAPTER 5

RESULTS

5.1 Outputs:

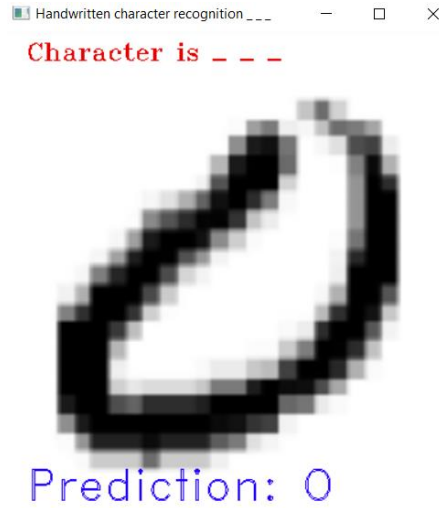


Fig 5.1 – “O” Character recognition output

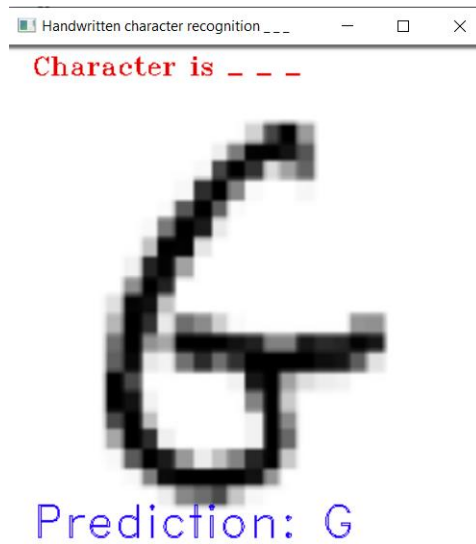


Fig 5.2 – “G” Character recognition output

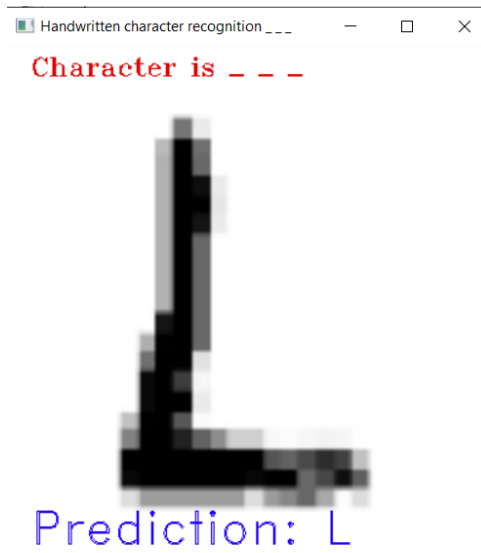


Fig 5.3 – “L” Character recognition output

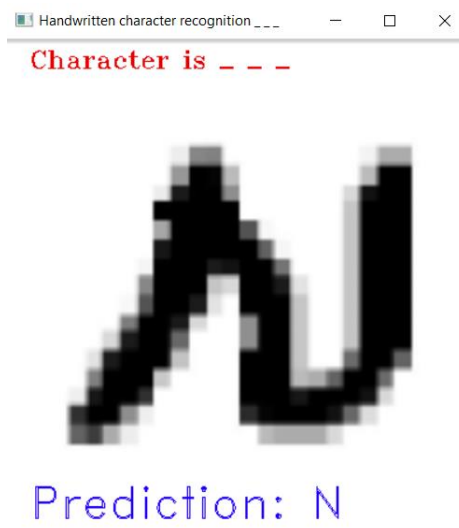


Fig 5.4 – “N” Character recognition output

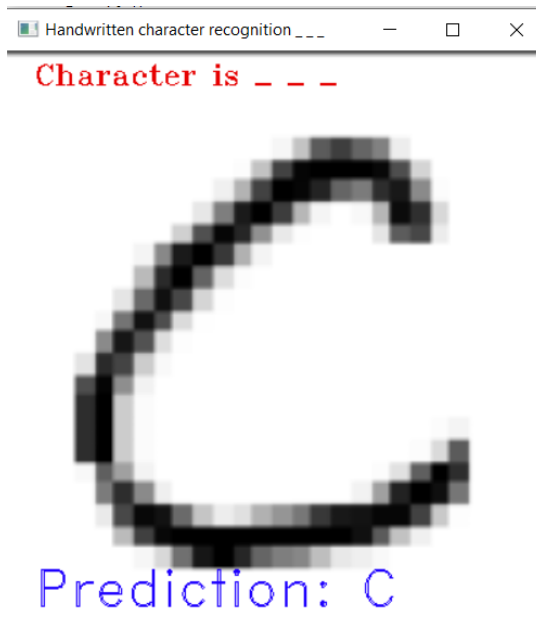


Fig 5.5 – “C” Character recognition output

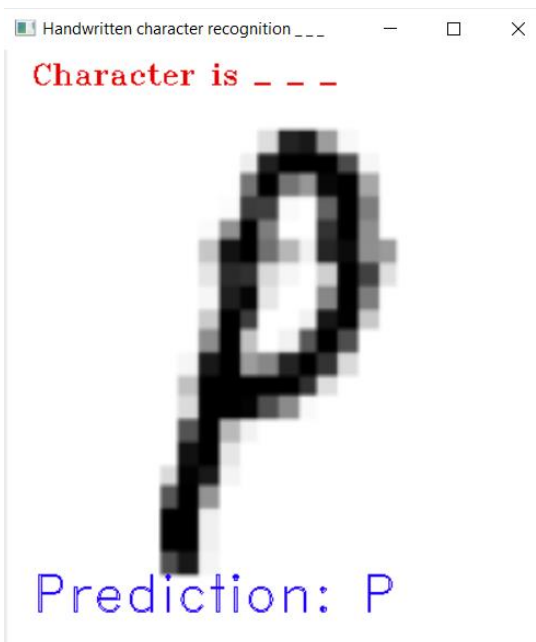


Fig 5.6 – “P” Character recognition output

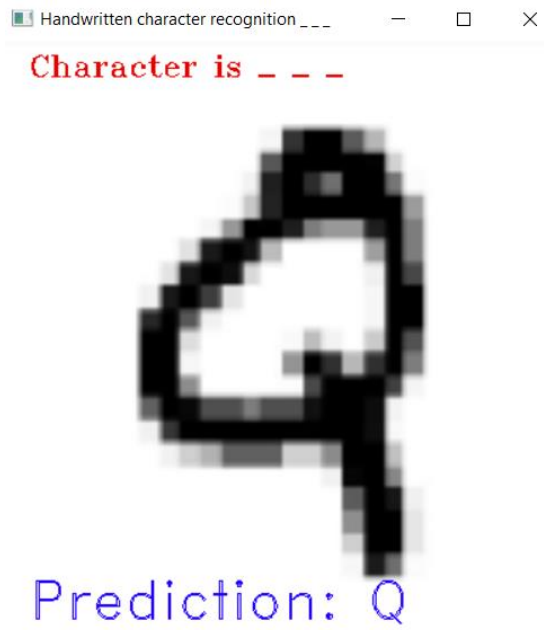


Fig 5.7 – “Q” Character recognition output

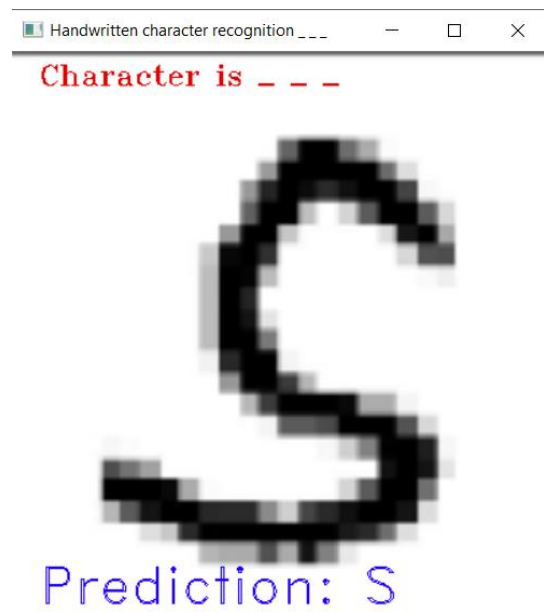


Fig 5.8 – “S” Character recognition output

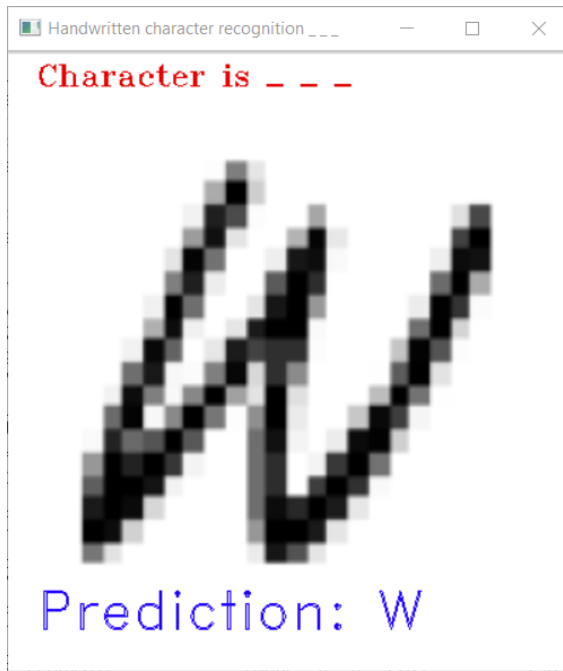
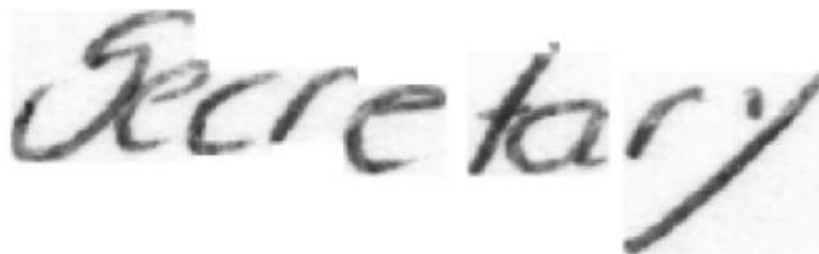
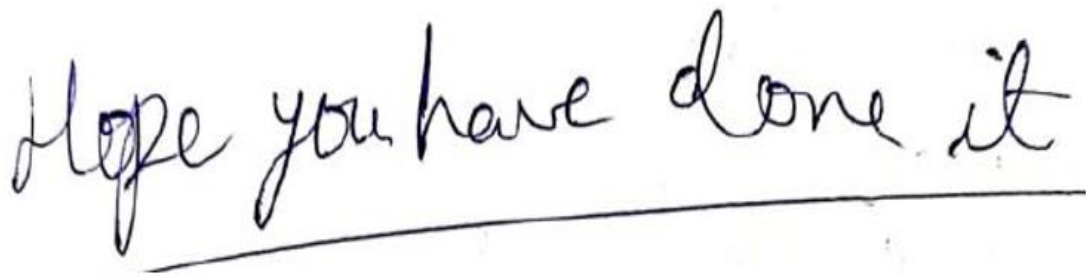


Fig 5.9 – “W” Character recognition output



Output is: Secretary

Fig 5.10 – “Secretary” text recognition output



Hope you have done it

Output is: Hope you have done it

Fig 5.11 – “Hope you have done it” text recognition output



Clothed leaf

Output is: Clothed leaf

Fig 5.12 – “Clothed leaf” text recognition output



today

Output is: today

Fig 5.13 – “today” text recognition output



nice

Output is: nice

Fig 5.14 – “nice” text recognition output

CHAPTER 6

CONCLUSION

As this model has a few boundaries to set, a sensible methodology is do a broad investigation of best qualities. As this can be a tedious and error-prone task, one can make utilize of some automated search/evaluation process. This project is deep learning approach using CNN to implement handwritten character recognition. And its aim is to recognize the character with the capability of CNN from A-Z Handwritten character data with a good accuracy in the model. And this got a very good score in accuracy. After upgradation and some modification is done in this project like character to text and multiple words recognition at the same time. For this we used CRNN that is Convolution recurrent neural network and with help of RNN with LSTM long short-term memory. As of now for the limited Output there is good accuracy but surely there should be some error too and that is the thing we have to work upon.

REFERENCES

- [1] Alex Graves, Santiago Fernández, Faustino Gomez, Jürgen Schmidhuber, Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, Proceedings of the 23rd international conference on Machine learning. 2006
- [2] Convolutional Neural Network Benchmarks: <https://github.com/jcjohnson/cnn-benchmarks>
- [3] Elie Kervat, Elliot Cuzzillo. Improving Off-line Handwritten Character Recognition with Hidden Markov Models.
- [4] Fabian Tschopp. Efficient Convolutional Neural Networks for Pixelwise Classification on Heterogeneous Hardware Systems
- [5] George Nagy. Document processing applications.
- [6] Mail encoding and processing system patent: <https://www.google.com/patents/US5420403>
- [7] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Neural Information Processing Systems (NIPS), 2015
- [8] H. Bunke¹, M. Roth¹, E.G. Schukat-Talamazzini. Offline Cursive Handwriting Recognition using Hidden Markov Models.
- [9] Kurzweil Computer Products. <http://www.kurzweiltech.com/kcp.html>
- [10] Oivind Due Trier, Anil K. Jain, Torfinn Taxt. Feature Extraction Methods for Character Recognition—A Survey. Pattern Recognition. 1996
- [11] Lisa Yan. Recognizing Handwritten Characters. CS 231N Final Research Paper.
- [12] K. Simonyan, A. Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition arXiv technical report, 2014
- [13] Tesseract Model: <https://github.com/tesseractocr/tesseract/wiki/TrainingTesseract-4.00>
- [14] Tesseract 4.0: <https://github.com/tesseractocr/tesseract/wiki/4.0-with-LSTM>
- [15] How many words are there in the English language?: <https://en.oxforddictionaries.com/explore/howmany-words-are-there-in-the-english-language>
- [16] Thodore Bluche, Jérôme Louradour, Ronaldo Messina. Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention.
- [17] T. Wang, D. Wu, A. Coates, A. Ng. "End-to-End Text Recognition with Convolutional Neural Networks" ICPR 2012.
- [18] U. Marti and H. Bunke. The IAM-database: An English Sentence Database for Off-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition, Volume 5, pages 39 - 46, 2002.

- [19] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, *Intelligent Signal Processing*, 306-351, IEEE Press, 2001 7
- [20] Zhang, X., He, K., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)
- [21] Plamondon R and Srihari S N 2000 *IEEE Trans. on Patt. Anal. and Mach. Intelligence* 22 63
- [22] Kato N, Suzuki M, Omachi S I, Aso H and Nemoto Y 1999 *IEEE Trans. on Patt. Anal. and Mach. Intelligence* 21 258
- [23] Islam M M, Tayan O, Islam M R, Islam M S, Nooruddin S, Kabir M N and Islam M R 2020 *IEEE Access* 8 166117
- [24] Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J and Chen T 2018 *Patt. Recognition* 77 354
- [25] Ptucha R, Such F P, Pillai S, Brockler F, Singh V and Hutkowski P 2019 *Patt. Recognition* 88 604
- [26]. G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.
- [27]. L. Eikvil, "Optical character recognition," *citeseer.ist.psu.edu/142042.html*, 1993.
- [28]. A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in neural information processing systems*, 2009, pp. 545–552.
- [29]. 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.
- [30]. S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving offline handwritten text recognition with hybrid hmm/ann models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 4, pp. 767–779, 2011.
- [31]. T. S. Gunawan, A. F. R. M. Noor, and M. Kartiwi, "Development of english handwritten recognition using deep neural network," 2018.
- [32]. C. Wu, W. Fan, Y. He, J. Sun, and S. Naoi, "Handwritten character recognition by alternately trained relaxation convolutional neural network," 2014 14th International Conference on Frontiers in Handwriting Recognition, pp. 291–296, 2014.
- [33]. F. Chollet et al., "Keras," <https://github.com/fchollet/keras>, 2015.
- [34]. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M.

Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015

[35] Kaggle Dataset <https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format>

APPENDIX

```
In [1]: from keras.datasets import mnist
import matplotlib.pyplot as plt
import cv2
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from keras.optimizers import SGD, Adam
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras.utils import to_categorical
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from keras.utils import np_utils
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
from sklearn.utils import shuffle
```

```
In [2]: # Read the data...
data = pd.read_csv(r"A_Z Handwritten Data.csv").astype('float32')
```

```
In [3]: # Split data the X - Our data , and y - the prdict Label
X = data.drop('0',axis = 1)
y = data['0']
```

```
In [4]: # Reshaping the data in csv file so that it can be displayed as an image...

train_x, test_x, train_y, test_y = train_test_split(X, y, test_size = 0.2)
train_x = np.reshape(train_x.values, (train_x.shape[0], 28,28))
test_x = np.reshape(test_x.values, (test_x.shape[0], 28,28))

print("Train data shape: ", train_x.shape)
print("Test data shape: ", test_x.shape)
```

```
Train data shape: (297960, 28, 28)
Test data shape: (74490, 28, 28)
```

```
In [5]: # Dictionary for getting characters from index values...
word_dict = {0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',13:'N',14:'O',15:'P',16:'Q',17:'R',18:'S',19:'T',20:'U',21:'V',22:'W',23:'X',24:'Y',25:'Z'}
```

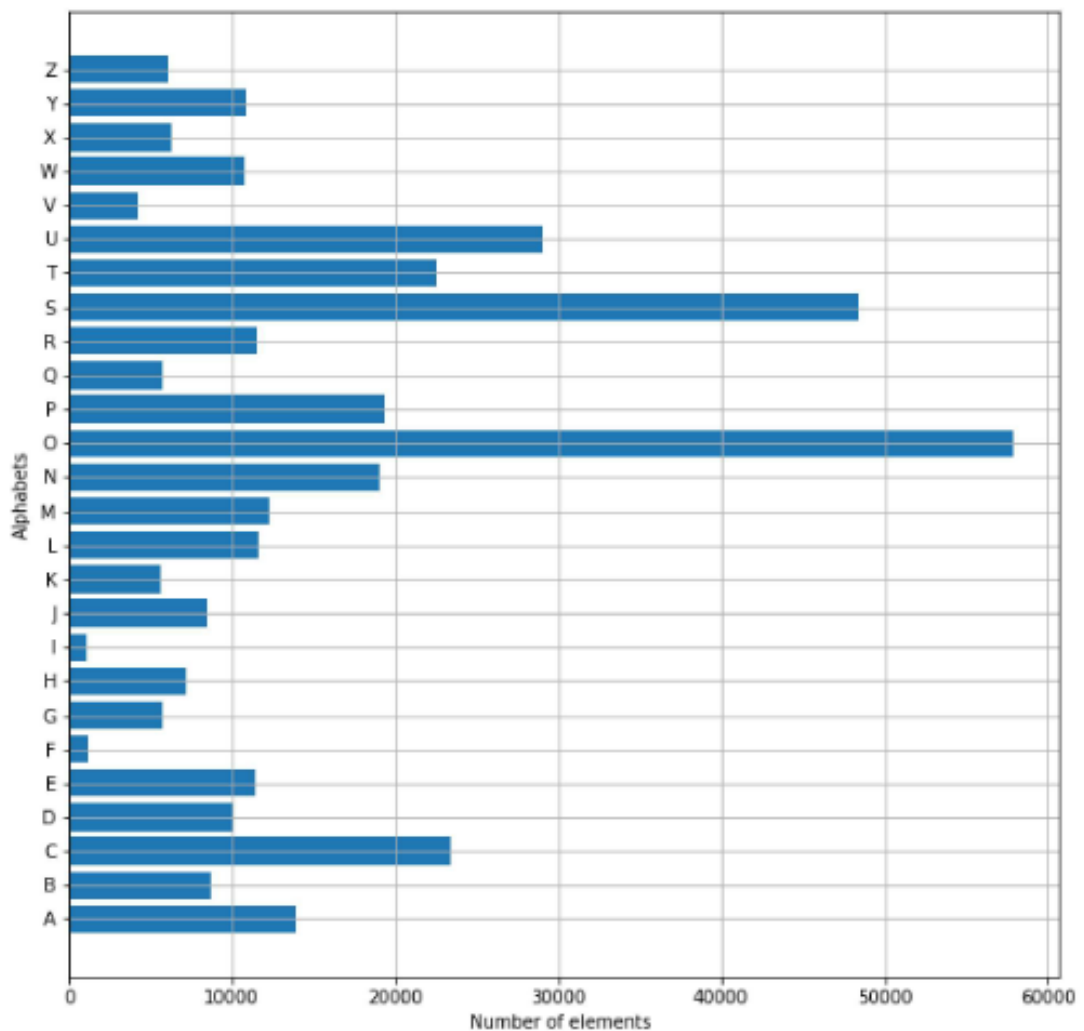
```
In [6]: # Plotting the number of alphabets in the dataset...
```

```
train_yint = np.int0(y)
count = np.zeros(26, dtype='int')
for i in train_yint:
    count[i] +=1

alphabets = []
for i in word_dict.values():
    alphabets.append(i)

fig, ax = plt.subplots(1,1, figsize=(10,10))
ax.barh(alphabets, count)

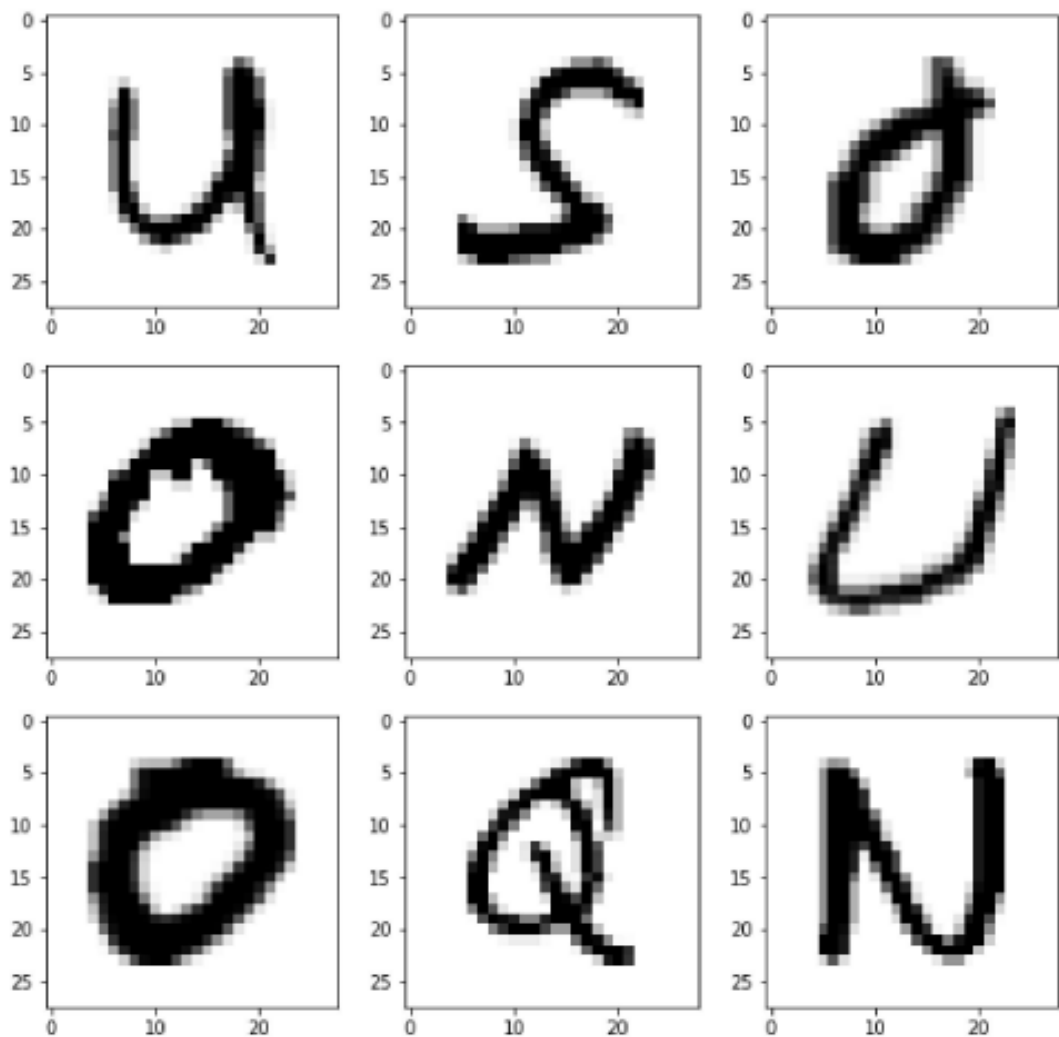
plt.xlabel("Number of elements ")
plt.ylabel("Alphabets")
plt.grid()
plt.show()
```




```
In [7]: #Shuffling the data ...
shuff = shuffle(train_x[:100])

fig, ax = plt.subplots(3,3, figsize = (10,10))
axes = ax.flatten()

for i in range(9):
    axes[i].imshow(np.reshape(shuff[i], (28,28)), cmap="Greys")
plt.show()
```



```
In [8]: #Reshaping the training & test dataset so that it can be put in the model...

train_X = train_x.reshape(train_x.shape[0],train_x.shape[1],train_x.shape[2],1)
print("New shape of train data: ", train_X.shape)

test_X = test_x.reshape(test_x.shape[0], test_x.shape[1], test_x.shape[2],1)
```

New shape of train data: (297960, 28, 28, 1)

```
In [9]: # Converting the labels to categorical values...

train_yOHE = to_categorical(train_y, num_classes = 26, dtype='int')
print("New shape of train labels: ", train_yOHE.shape)

test_yOHE = to_categorical(test_y, num_classes = 26, dtype='int')
print("New shape of test labels: ", test_yOHE.shape)
```

New shape of train labels: (297960, 26)

New shape of test labels: (74490, 26)

```
In [10]: # CNN model...

model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'sa
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = '\
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Flatten())

model.add(Dense(64,activation = "relu"))
model.add(Dense(128,activation = "relu"))

model.add(Dense(26,activation = "softmax"))

model.compile(optimizer = Adam(learning_rate=0.001), loss='categorical_crossentropy
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=
early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0

history = model.fit(train_X, train_yOHE, epochs=1, callbacks=[reduce_lr, early_st

model.summary()
model.save(r'model_hand.h5')
```

WARNING:tensorflow:From C:\Users\Bhupender Yadav\anaconda3\lib\site-packages \keras\backend\tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From C:\Users\Bhupender Yadav\anaconda3\lib\site-packages \keras\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 297960 samples, validate on 74490 samples
Epoch 1/1
297960/297960 [=====] - 413s 1ms/step - loss: 0.1679
- accuracy: 0.9547 - val_loss: 0.1029 - val_accuracy: 0.9720
Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_2 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_3 (Conv2D)	(None, 4, 4, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 128)	0

flatten_1 (Flatten)	(None, 512)	0
dense_1 (Dense)	(None, 64)	32832
dense_2 (Dense)	(None, 128)	8320
dense_3 (Dense)	(None, 26)	3354
=====		
Total params: 137,178		
Trainable params: 137,178		
Non-trainable params: 0		

```
In [11]: # Displaying the accuracies & losses for train & validation set...

print("The validation accuracy is :", history.history['val_accuracy'])
print("The training accuracy is :", history.history['accuracy'])
print("The validation loss is :", history.history['val_loss'])
print("The training loss is :", history.history['loss'])

The validation accuracy is : [0.9719962477684021]
The training accuracy is : [0.95470536]
The validation loss is : [0.10290976331905201]
The training loss is : [0.16793843845979098]
```

```
In [12]: #Making model predictions...

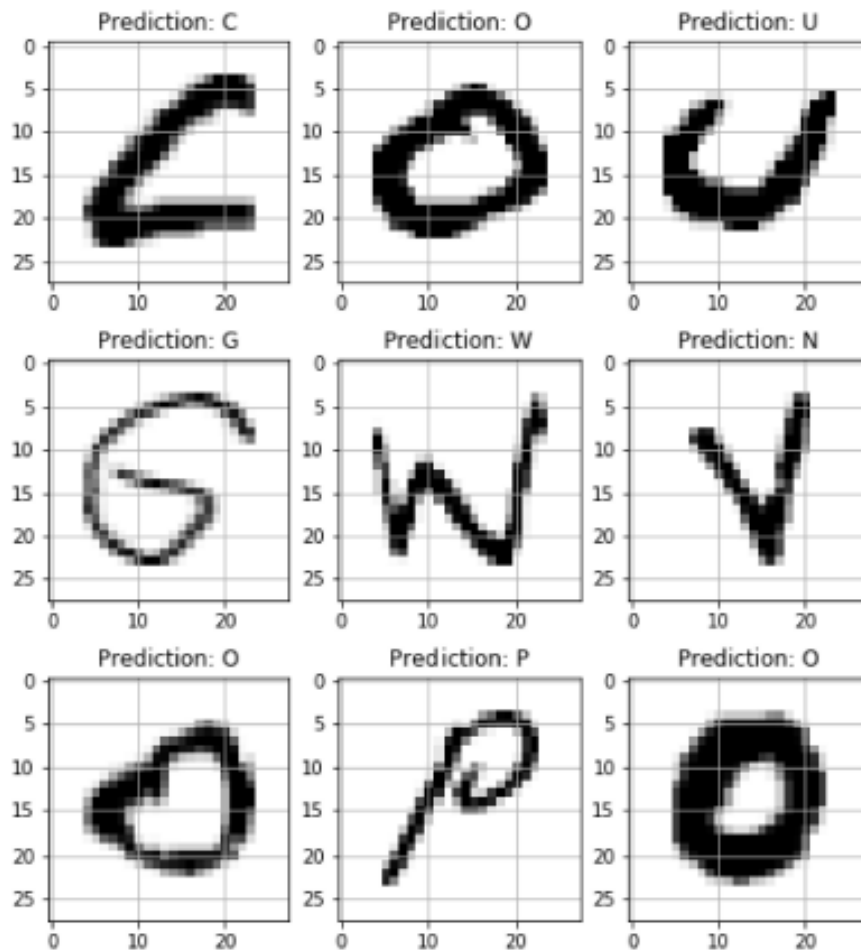
pred = model.predict(test_X[:9])
print(test_X.shape)

(74490, 28, 28, 1)
```

In [13]: # Displaying some of the test images & their predicted labels...

```
fig, axes = plt.subplots(3,3, figsize=(8,9))
axes = axes.flatten()

for i,ax in enumerate(axes):
    img = np.reshape(test_X[i], (28,28))
    ax.imshow(img, cmap="Greys")
    pred = word_dict[np.argmax(test_yOHE[i])]
    ax.set_title("Prediction: "+pred)
    ax.grid()
```



```

# Prediction on external image...

img = cv2.imread(r'img_o.jpg')
img_copy = img.copy()

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (400,440))

img_copy = cv2.GaussianBlur(img_copy, (7,7), 0)
img_gray = cv2.cvtColor(img_copy, cv2.COLOR_BGR2GRAY)
_, img_thresh = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY_INV)

img_final = cv2.resize(img_thresh, (28,28))
img_final = np.reshape(img_final, (1,28,28,1))

img_pred = word_dict[np.argmax(model.predict(img_final))]

cv2.putText(img, "Character is _ _ _", (20,25), cv2.FONT_HERSHEY_TRIPLEX, 0.7, c
cv2.putText(img, "Prediction: " + img_pred, (20,410), cv2.FONT_HERSHEY_DUPLEX, 1.
cv2.imshow('Handwritten character recognition _ _ _', img)

while (1):
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows()

```

```

def setupCNN(self):
    """ Create CNN Layers and return output of these Layers """

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

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

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

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

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

```



```

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

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

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

self.cnnOut4d = pool

```

```

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

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

    # BxTxH + BxTxH -> BxTx2H -> BxTx1x2H
    concat = tf.expand_dims(tf.concat([forward, backward], 2), 2)

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

```



```

def setupCTC(self):
    """ Create CTC Loss and decoder and return them """
    # BxTxC -> TxBxC
    self.ctcIn3dTBC = tf.transpose(self.rnnOut3d, [1, 0, 2])

    # Ground truth text as sparse tensor
    with tf.name_scope('CTC_Loss'):
        self.gtTexts = tf.SparseTensor(tf.placeholder(tf.int64, shape=[
            None, 2]), tf.placeholder(tf.int32, [None]), tf.placeholder(tf.int64, [2]))

        # Calculate loss for batch
        self.seqLen = tf.placeholder(tf.int32, [None])
        self.loss = tf.reduce_mean(tf.nn.ctc_loss(labels=self.gtTexts, inputs=self.ctcIn3dTBC, sequence_length=self.s
            ctc_merge_repeated=True, ignore_longer_outputs_than_inputs=True))
    with tf.name_scope('CTC_Decoder'):
        # Decoder: Best path decoding or Word beam search decoding
        if self.decoderType == DecoderType.BestPath:
            self.decoder = tf.nn.ctc_greedy_decoder(
                inputs=self.ctcIn3dTBC, sequence_length=self.seqLen)
        elif self.decoderType == DecoderType.BeamSearch:
            self.decoder = tf.nn.ctc_beam_search_decoder(inputs=self.ctcIn3dTBC, sequence_length=self.seqLen, beam_wi
        elif self.decoderType == DecoderType.WordBeamSearch:
            # Import compiled word beam search operation (see https://github.com/githubharald/CTCWordBeamSearch)
            word_beam_search_module = tf.load_op_library(
                './TFWordBeamSearch.so')

            # Prepare: dictionary, characters in dataset, characters forming words
            chars = codecs.open(FilePaths.wordCharList.txt, 'r').read()
            wordChars = codecs.open(
                FilePaths.fnWordCharList, 'r').read()
            corpus = codecs.open(FilePaths.corpus.txt, 'r').read()

            # Decoder using the "NGramsForecastAndSample": restrict number of (possible) next words to at most 20 w
            # decoder = word_beam_search_module.word_beam_search(tf.nn.softmax(ctcIn3dTBC, dim=2), 25, 'NGramsForecas

            # Decoder using the "Words": only use dictionary, no scoring: 0(1) mode of word beam search
            self.decoder = word_beam_search_module.word_beam_search(tf.nn.softmax(
                self.ctcIn3dTBC, dim=2), 25, 'Words', 0.0, corpus.encode('utf8'), chars.encode('utf8'), wordChars.enc

    # Return a CTC operation to compute the loss and CTC operation to decode the RNN output
    return self.loss, self.decoder

```

```

return self.loss, self.decoder

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

    return (sess, saver)

```

