

HANDWRITTEN CHARACTER RECOGNITION

Submitted in partial fulfillment of the requirements for

the award of

Bachelor of Engineering degree in Computer Science and Engineering

by

SAGAR KUMAR SINGH (Reg. No. 38110475)

RAHIM ALAM (Reg. No. 38110441)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600 119

MARCH 2022



SATHYABAMA



INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **S A G A R K U M A R S I N G H** (Reg. No. 3 8 110 4 7 5) and **R A H I M A L A M** (Reg. No. 3 8 110 4 4 1) who carried out the project entitled "**Handwritten character recognition**" under my supervision from **August 2021** to **March 2022**.

Internal Guide

Dr. L. SUJI HELEN M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **SAGAR KUMAR SINGH** hereby declare that the Project Report entitled “**Handwritten Character Recognition**” is done by me under the guidance **Dr. L. SUJI HELEN, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. L.SUJI HELEN, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In today's world advancement in sophisticated scientific techniques is pushing further the limits of human outreach in various fields of technology. One such field is the field of character recognition commonly known as OCR (Optical Character Recognition). In this fast paced world there is an immense urge for the digitization of printed documents and documentation of information directly in digital form. And there is still some gap in this area even today. OCR techniques and their continuous improvisation from time to time is trying to fill this gap. This project is about devising an algorithm for recognition of hand written characters also known as HCR (Handwritten Character Recognition) leaving aside types of OCR that deals with recognition of computer or typewriter printed characters. A novel technique is proposed for recognition English language characters using Artificial Neural Network including the schemes of feature extraction of the characters and implemented. The persistency in recognition of characters by the Artificial neural network was found to be more than 90% of times.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	V
	LIST OF FIGURES	VIII
1.	INTRODUCTION	9
	1.1 INTRODUCTION TO MODELS	9
	1.2 OBJECTIVE OF THE PROJECT	9
2.	LITERATURE SURVEY	10
	2.1 LITERATURE SURVEY	10
3.	METHODOLOGY	12
	3.1 ARTIFICIAL NEURAL NETWORK	12
	3.1.1 CREATING AND TRAINING OF NETWORK	12
	3.1.2 ARCHITECTURE OF ARTIFICIAL NEURAL NETWORK	12
	3.1.3 ADVANTAGES OF ARTIFICIAL NEURAL NETWORK (ANN)	14
	3.2 CONVOLUTION NEURAL NETWORK	15
	3.3 PRE-PROCESSING OF SAMPLE IMAGE	15
	3.3.1 GREY-SCALING OF RGB IMAGE	16
	3.3.2 BINARIZATION	16
	3.3.3 INVERSION	17
	3.4 FEATURE EXTRACTION	19
	3.4.1 STATISTICAL FEATURE EXTRACTION	19

	3.4.2 STRUCTURAL FEATURE EXTRACTION	19
	3.4.2.1 INDEXING AND LABELLING	20
	3.4.2.2 BOXING AND CROPPING	20
	3.4.2.3 RESHAPING AND RESIZING	20
	3.5 ADAM OPTIMIZATION ALGORITHM	20
	3.5.1 HOW DOES ADAM WORK	21
	3.6 WHAT IS CROSS ENTROPY	23
	3.6.1 CROSS ENTROPY AS A LOSS FUNCTION	25
	3.7 WORK PLAN	27
4.	RESULTS AND DISCUSSIONs	29
5.	CONCLUSION AND FUTURE WORK	34
	5.1 CONCLUSION	34
	5.2 FUTURE WORK	34
	REFERENCES	35
	APPENDIX	36

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.1	Architecture of an artificial neural network	13
3.2	RGB to Greyscale conversion of alphabet A.	16
3.3	Inversion	19
3.4	Architecture Diagram of working system	27
4.1	Visualization of content of dataset	29
4.2	Prediction on Test data.	30
4.3	Prediction on External Image 1	31
4.4	Prediction on External Image 2	32
4.5	Prediction on External Image 3	33
4.6	Prediction on External Image 4	34

CHAPTER - 1

INTRODUCTION

1.1 INTRODUCTION OF MODEL

This project, 'Handwritten Character Recognition' is a software algorithm project to recognize any hand written character efficiently on computer with input in an image format . Character recognition, usually abbreviated to optical character recognition or shortened OCR, is the mechanical or electronic translation of images of handwritten, typewritten or printed text (usually captured by a scanner) into machine-editable text. It is a field of research in pattern recognition, artificial intelligence and machine vision. Though academic research in the field continues, the focus on character recognition has shifted to implementation of proven techniques. Optical character recognition is a scheme which enables a computer to learn, understand, improvise and interpret the written or printed character in their own language, but present correspondingly as specified by the user. Optical Character Recognition uses the image processing technique to identify any character computer/typewriter printed or hand written. A lot of work has been done in this field. But a continuous improvisation of OCR techniques is being done based on the fact that algorithm must have higher accuracy of recognition, higher persistency in number of times of correct prediction and increased execution time. The idea is to device efficient algorithms which get input in digital image format. After that it processes the image for better comparison. Then after the processed image is compared with already available set of font images. The last step gives a prediction of the character in percentage accuracy.

1.2 Objective of the Project

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.

CHAPTER - 2

Literature Survey

2.1 Literature Survey

Research in the region of word recognition, being done from Grimsdale in the year 1959 is soonest endeavor to perceive the handwritten character. This mid-sixty research exhibited the utilization of examination by combination strategy being proposed by the eden in 1968. He demonstrated that every single handwritten character is limited to number of schematic highlights. This hypothesis was later utilized as a part of almost all strategies for auxiliary methodologies in the region of character recognition. K. Gaurav and Bhatia P. K [2], proposed different prehandling systems being associated with the recognition of the characters. The procedure took a shot at the various types of pictures from a basic picture-based report to a hue and changed forces including foundation. Different systems of pre-handling and standardization like skew remedy, differentiate evacuation, commotion expulsion and numerous other upgrade procedures were recommended. They reached the decision that a solitary procedure can't be connected for preprocessing the picture. Yet additionally there were a few disparities that utilizing every one of these systems likewise can't give the best exactness comes about. Salvador España-Boquera [3], The analysts proposed the utilization of hybrid or half plus half concealed markov show (HMM) to perceive the handwritten content in disconnected mode. The optical model's basic part was prepared with markov chain procedure and a multilayer perceptron was likewise used to gauge the probabilities.

In [4], to perceive the disconnected handwritten numerals of six prominent Indian language, a changed quadratic classifier is utilized. A similar paper likewise manages perceiving the English letters in order. For both of these, a multilayer perceptron was utilized and Boundary following and Fourier descriptors were utilized for the component extraction. By examining the shape and looking at their highlights,

the characters were identified. Also, to decide the quantity of concealed layers, back spread system was utilized. With this very calculation, a recognition rate of 94% have been accounted for with less preparing time. R. Bajaj, S. Chaudhari, L. Dey, et al [5], for grouping the Devanagari numerals, distinctive highlights like clear part, thickness and minute highlights were utilized. Additionally, to increase the recognition capacity, the paper proposes multi classifier unwavering quality for handwritten Devanagari numerals. Sandhya Arora in [6], In this paper specifically four highlights like shadow, histogram of chain code crossing point and horizontal line fitting highlights being portrayed. Among these highlights the shadow was registered all around for picture character, the rest three were processed by partitioning the character picture into the distinctive sections. In the one useful execution utilizing the dataset of 4900 examples demonstrated the exactness rate of 90.8 % for Devanagari handwritten characters. Nafiz Arica at al. [7] This paper gave the technique because of which it was less demanding to maintain a strategic distance from the preprocessing stage along these lines lessening the loss of imperative data. The best one proposed was calculation of capable division. What's more, the different strategies supporting this calculation were utilizing neighborhood maxima and minima, additionally other, for example, stroke tallness which turned out to be ideal and furthermore character limit. What's more, these were altogether connected on a grayscale picture. Utilizing this approach, superfluous division was decreased bit by bit. Alongside that, the paper additionally proposed another model called shrouded markov demonstrate (HMM) preparing for estimation of worldwide and highlight space parameters alongside estimation of model parameters. Additionally, to rank the individual characters and furthermore to get the shape data, this preparation show was utilized. Additionally, by utilizing the one-dimensional portrayal of a 2-D character picture tremendously builds the energy of HMM for shape perceiving. In [8], a technique was proposed to perceive the individually Tamil written character by utilizing the grouping in the strokes. Principally a strokes' format or shape-based portrayal is utilized spoken to as a string of shape highlights. Utilizing this strategy, the unrecognized stroke was perceived by contrasting it and a dataset of strokes by the string coordinating method in an adaptable mode. Utilizing this, an individual character was perceived

by distinguishing every one of the strokes and its segments.

CHAPTER 3

METHODOLOGY

3.1 Artificial Neural Network

An early phase of Neural Network was developed by Warren McCulloch and Walter Pitts in 1943 which was a computational model based on Mathematics and algorithm. This model paved the way for research which was focused on the application of Neural Networks in Artificial Intelligence. Artificial neural network is basically a mesh of large number of interconnected cells. The arrangement of cells are such that each cell receives an input and drives an output for subsequent cells. The diagram below is a block diagram that depicts the structure and work flow of a created Artificial Neural Network. The neurons are interconnected with each other in a serial manner. The network consist of a number of hidden layers depending upon the resolution of comparison of inputs with the dataset.

3.1.1 Creating and training of network

In case of character recognition we have to create a 2D vector of character images which can be fed to the network as ideal set of input variables. In our case there is a total of 26 capital English letters which we are to recognize. Below is a set of characters written in binary form of 7x5 sized matrix of 26 capital English letters:

3.1.2 The architecture of an artificial neural network:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Let us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:

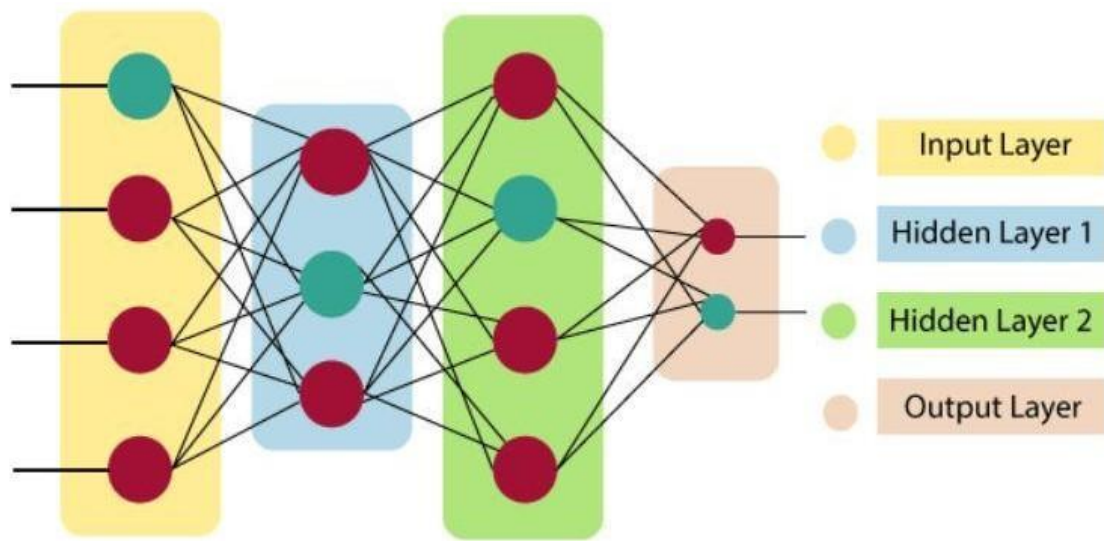


Fig.3.1 Architecture of an artificial neural network

Input Layer:

As the name suggests, it accepts inputs in several different formats provided by the programmer.

Hidden Layer:

The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

Output Layer:

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^n W_i * X_i + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

3.1.3 Advantages of Artificial Neural Network (ANN)

Parallel processing capability:

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

Storing data on the entire network:

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

Capability to work with incomplete knowledge:

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

Having a memory distribution:

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

Having fault tolerance:

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

3.1 Convolutional Neural Network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels that scan the hidden layers and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, Image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include varying the weights as the loss function gets minimized while randomly trimming connectivity. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in the filters. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters or convolution kernels that in traditional algorithms are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

3.2 Pre-processing of sample image

Pre-processing of the sample image involves few steps that are mentioned as follows:

3.2.1 Grey-scaling of RGB image

Grey-scaling of an image is a process by which an RGB image is converted into a black and white image. This process is important for Binarization as after grey scaling of the image, only shades of grey remains in the image, binarization of such image is efficient.

An RGB image can be viewed as three images(a red scale image, a green scale image and a blue scale image) stacked on top of each other. In MATLAB, an RGB image is basically a $M \times N \times 3$ array of colour pixel, where each colour pixel is a triplet which corresponds to red, blue and green colour component of RGB image at a specified spatial location.

Similarly, A Grayscale image can be viewed as a single layered image. In MATLAB, a grayscale image is basically $M \times N$ array whose values have been scaled to represent intensities.

In MATLAB, there is a function called `rgb2gray()` is available to convert RGB image to grayscale image. Here we will convert an RGB image to grayscale image without using `rgb2gray()` function.

Our key idea is to convert an RGB image pixel which a triplet value corresponding to red, blue and green colour component of an image at a specified spatial location to a single value by calculating a weighted sum of all three colour component.

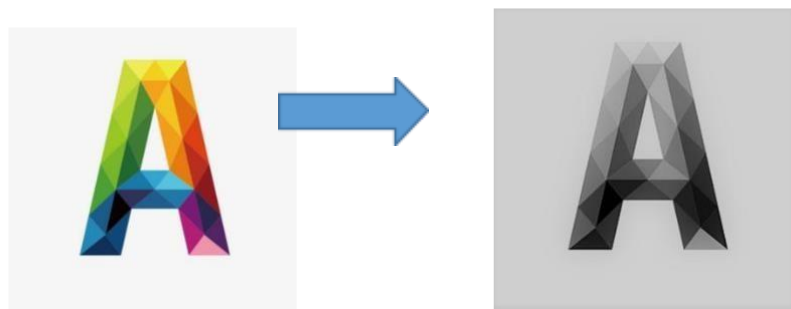


Fig.3.2 RGB to Greyscale conversion of alphabet A.

3.2.2 Binarization

Binarization of an image converts it into an image which only have pure black and pure white pixel values in it. Basically during binarization of a grey-scale image, pixels with intensity lower than half of the full intensity value gets a zero value converting them into black ones. And the remaining pixels get a full intensity value converting it into white pixels.

Binarization is the process converting a multi-tone image into a bi-tonal image. In the case of document images, it is typical to map foreground text pixels to black and the rest of the image (background) to white. In many applications, binarization is a critical preprocessing step and helps facilitate other document processing tasks such as layout analysis and character recognition. In such pipelines, the quality of the binarization can greatly affect system performance, as errors made in the binarization step can propagate to downstream tasks. As a standalone application, binarization can serve as a noise removal process to increase document readability. The file size of binary images is often orders of magnitudes smaller than the original gray or color images, which makes them cheaper to store on disk. Additionally, with the rise of digital archives, file size can become a concern as large numbers of images are viewed over the Internet. If a person can still recognize the text in the binary images, then this compression can be obtained with virtually no loss in semantic image content.

In the last decade, a tremendous amount of progress has been made in the field of historical document binarization. In 2009, the first Document Image Binarization Contest (DIBCO) introduced the first dataset of real degraded images that have ground truth annotations at the pixel level [40]. This enabled a standardized evaluation procedure that allowed for direct comparison between algorithms. This spurred research in the field and the creation of more datasets in new application domains.

3.2.3 Inversion

Inversion is a process in which each pixel of the image gets a colour which is the inverted colour of the previous one. This process is the most important one because any character on a sample image can only be extracted efficiently if it contains only one colour which is distinct from the background colour. Note that it is only required if the objects we have to identify if of darker intensity on a lighter background.

The flow chart shown below illustrates the physical meaning of the processes that are mentioned above:

RGB => Grey-scaling => Binarization => Inversion

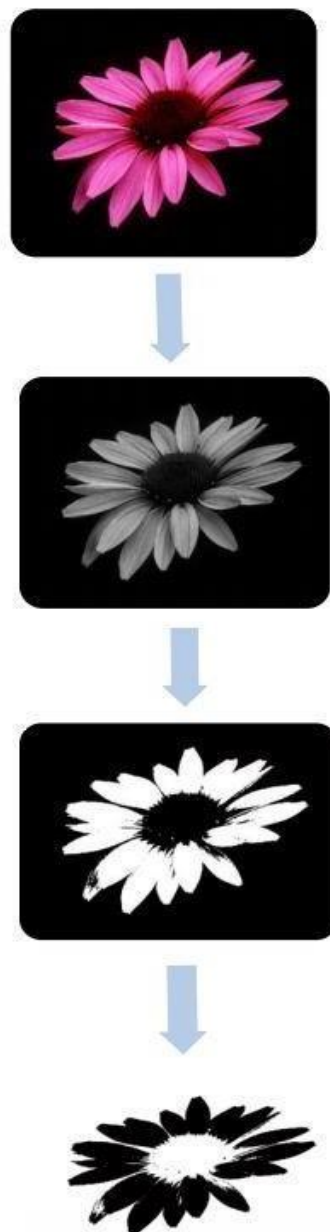


Fig.3.3 Inversion

3.3 FEATURE EXTRACTION

Features of a character depicts the morphological and spatial characteristics in the image. Feature extraction is a method of extracting of features of characters from the sample image. Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for

processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. Feature extraction is the name for methods that select and /or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.

Why this is Useful:

The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

There are basically two types of feature extraction:

- Statistical feature extraction

- Structural feature extraction

3.3.1 Statistical feature extraction

In this type of extraction the extracted feature vector is the combination of all the features extracted from each character. The associated feature in feature vector of this type of extraction is due to the relative positions of features in character image matrix.

3.3.2 Structural feature extraction

This is a primitive method of feature extraction which extracts morphological features of a character from image matrix. It takes into account the edges, curvature, regions, etc. This method extracts the features of the way character are written on image matrix. The different methods used for feature extraction are :-

- Piecewise

- linear regression

- Curve-fitting

Zoning

Chain code, etc.

The functions that are used in feature extraction are:

3.4.2.1 Indexing and labelling

This is a process by which distinct characters in an image are indexed and labelled in an image. Thus helps in classification of characters in image and makes feature extraction of characters simple.

3.4.2.2 Boxing and Cropping

This is a process of creating a boundary around the characters identified in an image. This helps by making cropping of characters easier. After boxing the characters are cropped out for storing them as input variables for recognition.

3.4.2.3 Reshaping and Resizing

Reshaping is done to change the dimensions of the acquired character in desired shape. Resizing is done to reduce the size of characters to a particular minimum level.

3.4 Adam Optimization Algorithm

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

Adam was presented by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 ICLR paper (poster) titled “Adam: A Method for

Stochastic Optimization“. I will quote liberally from their paper in this post, unless stated otherwise.

The algorithm is called Adam. It is not an acronym and is not written as “ADAM”.

When introducing the algorithm, the authors list the attractive benefits of using Adam on non-convex optimization problems, as follows:

- Straightforward to implement.
- Computationally efficient.
- Little memory requirements.
- Invariant to diagonal rescale of the gradients.
- Well suited for problems that are large in terms of data and/or parameters.
- Appropriate for non-stationary objectives.
- Appropriate for problems with very noisy/or sparse gradients.
- Hyper-parameters have intuitive interpretation and typically require little tuning.

3.4.1 How does ADAM work

Adam is different to classical stochastic gradient descent.

Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training.

A learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.

The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

The authors describe Adam as combining the advantages of two other extensions of stochastic gradient descent. Specifically:

- **Adaptive Gradient Algorithm** (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).
- **Root Mean Square Propagation** (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

Adam realizes the benefits of both AdaGrad and RMSProp.

Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).

Specifically, the algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters β_1 and β_2 control the decay rates of these moving averages.

The initial value of the moving averages and β_1 and β_2 values close to 1.0 (recommended) result in a bias of moment estimates towards zero. This bias is overcome by first calculating the biased estimates before then calculating bias-corrected estimates.

3.5 What is Cross Entropy

Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events.

You might recall that **information** quantifies the number of bits required to encode and transmit an event. Lower probability events have more information, higher probability events have less information.

In information theory, we like to describe the “*surprise*” of an event. An event is more surprising the less likely it is, meaning it contains more information.

- **Low Probability Event** (*surprising*): More information.

- **Higher Probability Event** (*unsurprising*): Less information.

Information $h(x)$ can be calculated for an event x , given the probability of the event $P(x)$ as follows:

- $h(x) = -\log(P(x))$

Entropy is the number of bits required to transmit a randomly selected event from a probability distribution. A skewed distribution has a low entropy, whereas a distribution where events have equal probability has a larger entropy.

A skewed probability distribution has less “surprise” and in turn a low entropy because likely events dominate. Balanced distributions are more surprising and turn have higher entropy because events are equally likely.

- **Skewed Probability Distribution** (*unsurprising*): Low entropy.
- **Balanced Probability Distribution** (*surprising*): High entropy.

Entropy $H(x)$ can be calculated for a random variable with a set of x in X discrete states and their probability $P(x)$ as follows:

- $H(X) = -\sum_{x \in X} P(x) * \log(P(x))$

If you would like to know more about calculating information for events and entropy for distributions see this tutorial:

- A Gentle Introduction to Information Entropy

Cross-entropy builds upon the idea of entropy from information theory and calculates the number of bits required to represent or transmit an average event from one distribution compared to another distribution.

the cross entropy is the average number of bits needed to encode data coming from a source with distribution p when we use model q

The intuition for this definition comes if we consider a target or underlying probability distribution P and an approximation of the target distribution Q , then the cross-entropy of Q from P is the number of additional bits to represent an event using Q instead of P .

The cross-entropy between two probability distributions, such as Q from P , can be

stated formally as:

- $H(P, Q)$

Where $H()$ is the cross-entropy function, P may be the target distribution and Q is the approximation of the target distribution.

Cross-entropy can be calculated using the probabilities of the events from P and Q , as follows:

- $H(P, Q) = - \sum_{x \in X} P(x) * \log(Q(x))$

Where $P(x)$ is the probability of the event x in P , $Q(x)$ is the probability of event x in Q and $\log$ is the base-2 logarithm, meaning that the results are in bits. If the base-e or natural logarithm is used instead, the result will have the units called nats.

This calculation is for discrete probability distributions, although a similar calculation can be used for continuous probability distributions using the integral across the events instead of the sum.

The result will be a positive number measured in bits and will be equal to the entropy of the distribution if the two probability distributions are identical.

3.5.1 Cross-Entropy as a Loss Function

Cross-entropy is widely used as a loss function when optimizing classification models.

Two examples that you may encounter include the logistic regression algorithm (a linear classification algorithm), and artificial neural networks that can be used for classification tasks.

Classification problems are those that involve one or more input variables and the prediction of a class label.

Classification tasks that have just two labels for the output variable are referred to as binary classification problems, whereas those problems with more than two

labels are referred to as categorical or multi-class classification problems.

- **Binary Classification:** Task of predicting one of two class labels for a given example.
- **Multi-Class Classification:** Task of predicting one of more than two class labels for a given example.

We can see that the idea of cross-entropy may be useful for optimizing a classification model.

Each example has a known class label with a probability of 1.0, and a probability of 0.0 for all other labels. A model can estimate the probability of an example belonging to each class label. Cross-entropy can then be used to calculate the difference between the two probability distributions.

As such, we can map the classification of one example onto the idea of a random variable with a probability distribution as follows:

- **Random Variable:** The example for which we require a predicted class label.
- **Events:** Each class label that could be predicted.

In classification tasks, we know the target probability distribution P for an input as the class label 0 or 1 interpreted as probabilities as “*impossible*” or “*certain*” respectively. These probabilities have no surprise at all, therefore they have no information content or zero entropy.

Our model seeks to approximate the target probability distribution Q .

In the language of classification, these are the actual and the predicted probabilities, or y and $\hat{y}$.

- **Expected Probability (y):** The known probability of each class label for an example in the dataset (P).
- **Predicted Probability ($\hat{y}$):** The probability of each class label an example predicted by the model (Q).

We can, therefore, estimate the cross-entropy for a single prediction using the cross-entropy calculation described above; for example.

- $H(P, Q) = - \sum_{x \in X} P(x) * \log(Q(x))$

Where each x in X is a class label that could be assigned to the example, and $P(x)$ will be 1 for the known label and 0 for all other labels.

The cross-entropy for a single example in a binary classification task can be stated by unrolling the sum operation as follows:

- $H(P, Q) = - (P(\text{class0}) * \log(Q(\text{class0})) + P(\text{class1}) * \log(Q(\text{class1})))$

You may see this form of calculating cross-entropy cited in textbooks.

If there are just two class labels, the probability is modeled as the Bernoulli distribution for the positive class label. This means that the probability for class 1 is predicted by the model directly, and the probability for class 0 is given as one minus the predicted probability, for example:

- **Predicted $P(\text{class0}) = 1 - \hat{y}$**
- **Predicted $P(\text{class1}) = \hat{y}$**

When calculating cross-entropy for classification tasks, the base-e or natural logarithm is used. This means that the units are in nats, not bits.

We are often interested in minimizing the cross-entropy for the model across the entire training dataset. This is calculated by calculating the average cross-entropy across all training examples.

3.7 Work Plan

The imported data set will be reshaped to 28x28 pixel size and classified into train data and test data. We initiate a word dictionary containing all English alphabet mapped to integer values. We visualize the imported dataset in graph form which indicates how many images are present of each alphabet.

Shuffling of all the data happens and then some random images are displayed from train dataset into a 9x9 matrix.

Reshaping of the data into categorical values for making it in the correct format for the CNN.

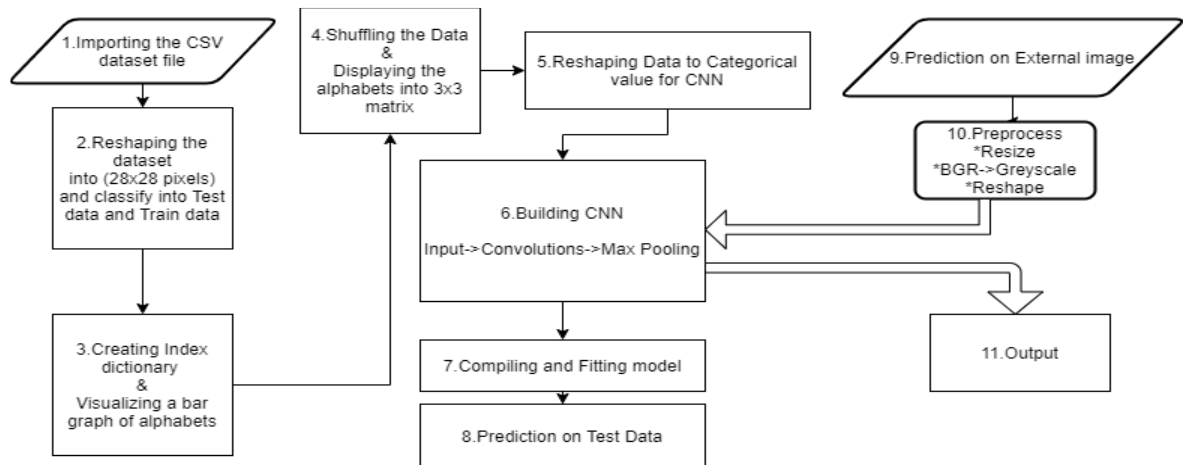


Fig.3.4 Architecture Diagram of working system

Building of CNN , the input image is passed through many convolution. This process is called feature extraction. When the image is matched with features its further send towards for pooling. This pooling process reduces the size of the image. Now the output is flattened into a vector form and further passed to dense layer matches the image and returns the index value for matching into word dictionary.

The model is optimized by using Adam's optimizing function and categorical cross-entropy loss function. Adam's optimizer is the extension of stochastic gradient descent which is good for compiling the model where the dataset is very large and chances of availability of redundant data.

The model now predicts the test data from the dataset into 3x3 plots.

After that we can predict image of external data. The external data is imported by using OpenCV which takes BGR image. For processing it into model we convert it to greyscale and resize it to 28x28 pixels and reshape the image to categorical values. This is fed to CNN model and it predicts the image.

CHAPTER 4

RESULTS AND DISCUSSIONS

“HCR Using Neural Network ” is aimed at recognizing the handwritten characters. The “Handwritten Character Recognition System ”is implemented using a neural network.in this system original image is converted into gray scale image then After gray scaling image is converted in black and white and segmented form. After preprocessing and segmentation operation system show final output.

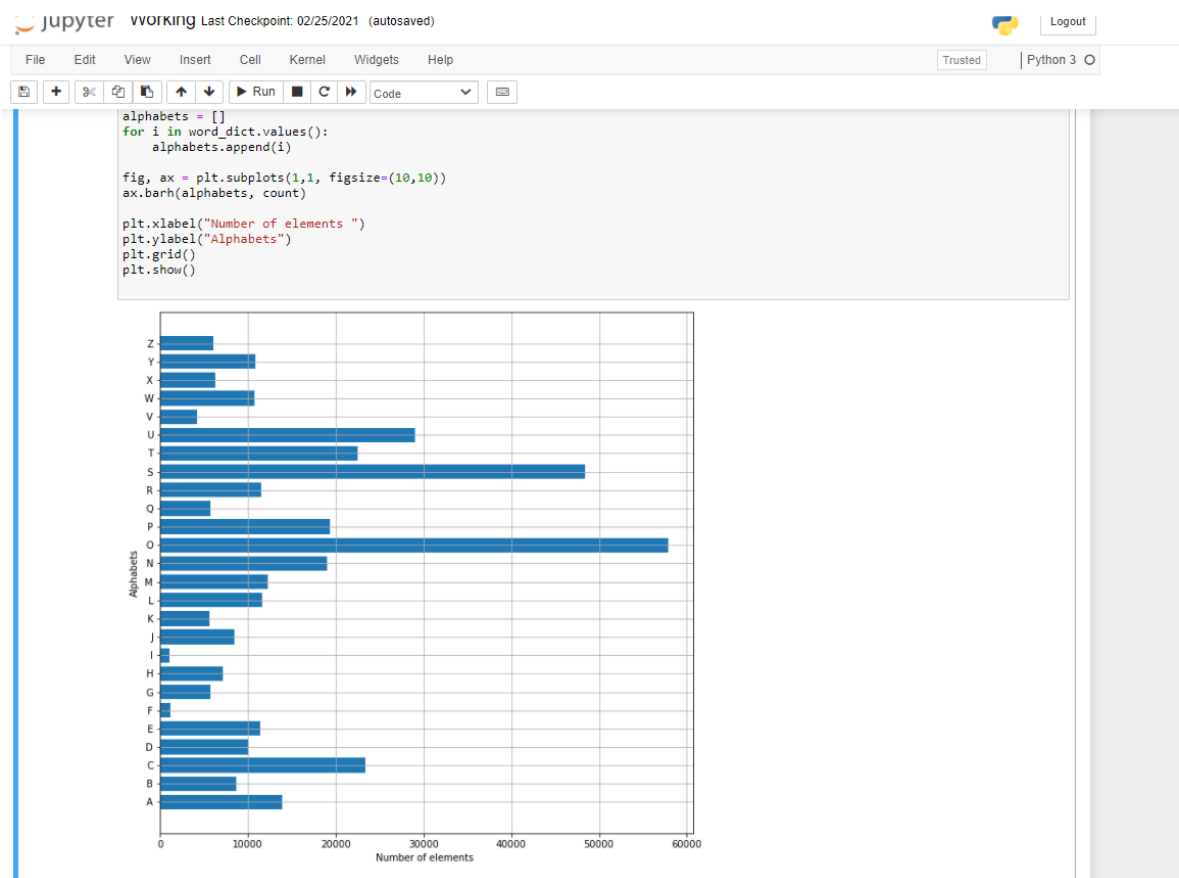


Fig.4.1 Visualization of content of dataset

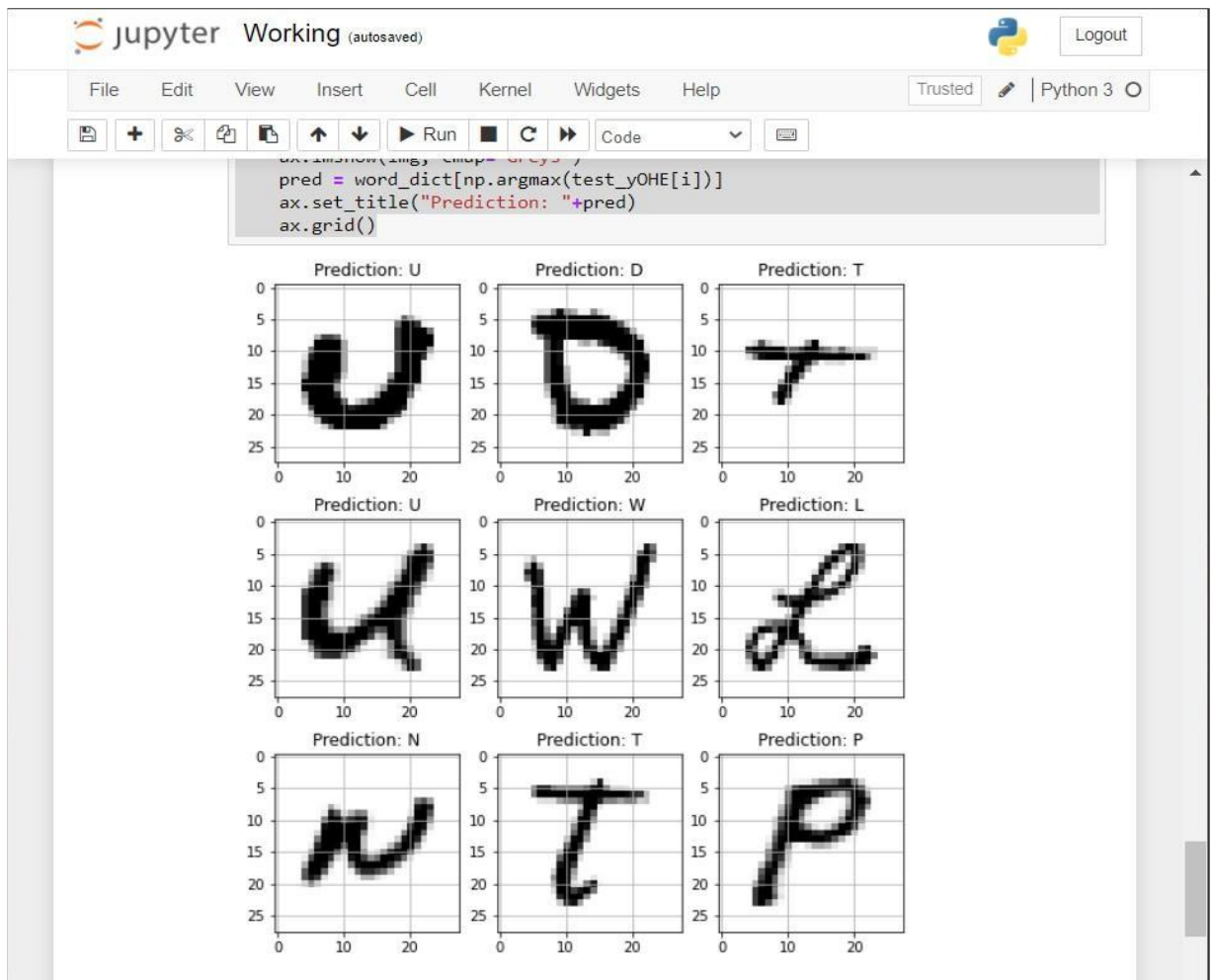


Fig.4.2 Prediction on Test data

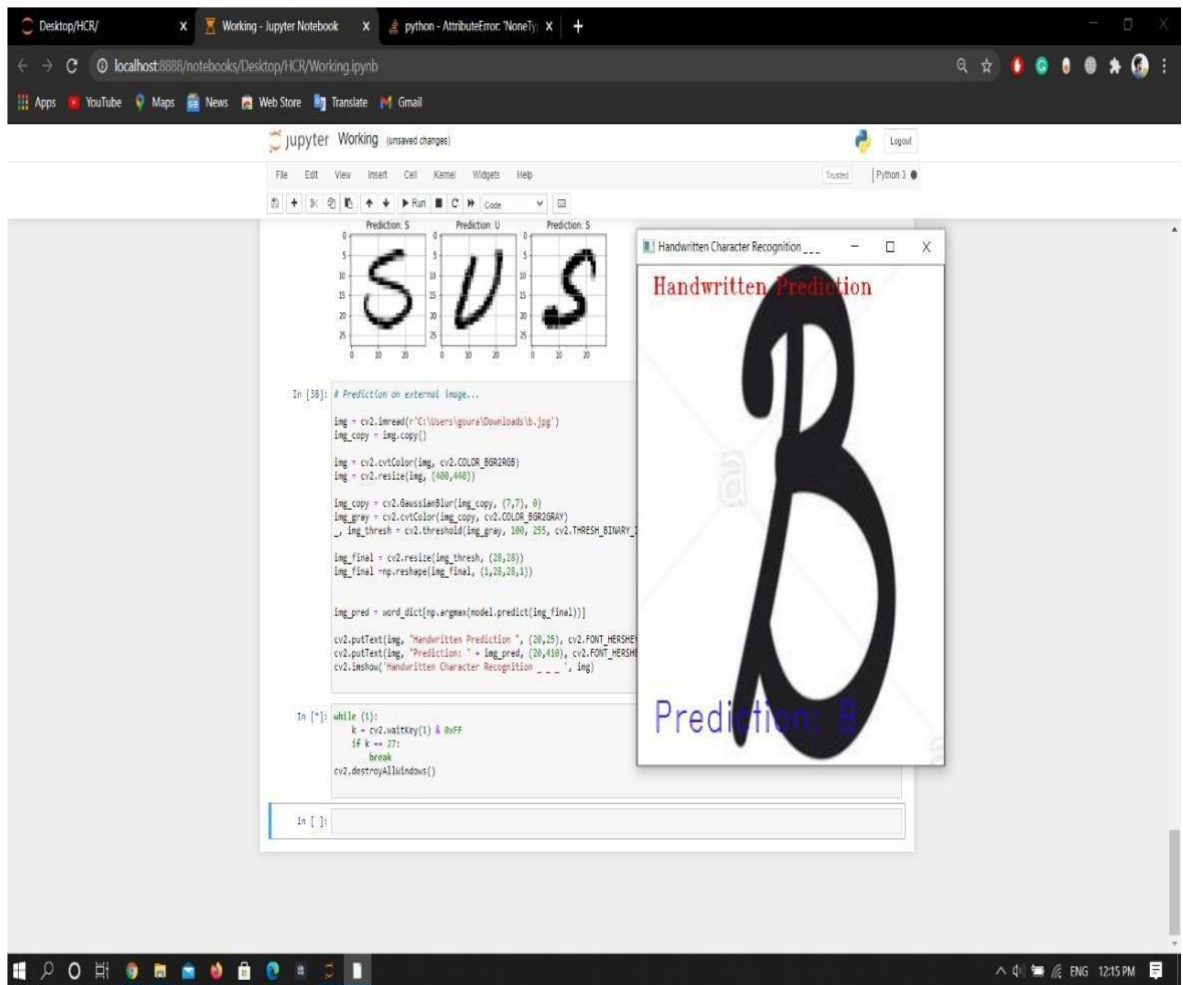


Fig.4.3 Prediction on External Image 1



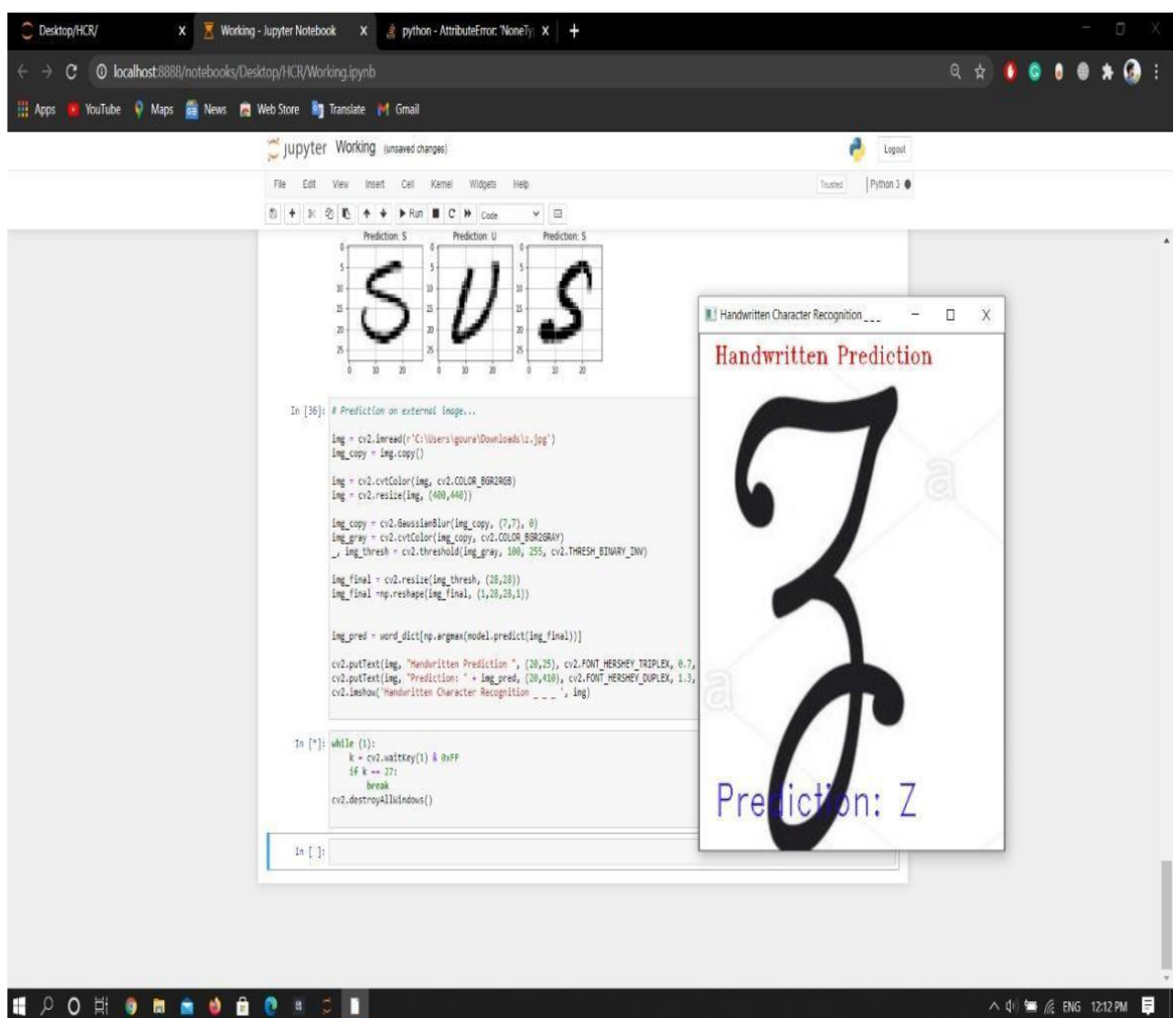


Fig.4.5 Prediction on External Image 3

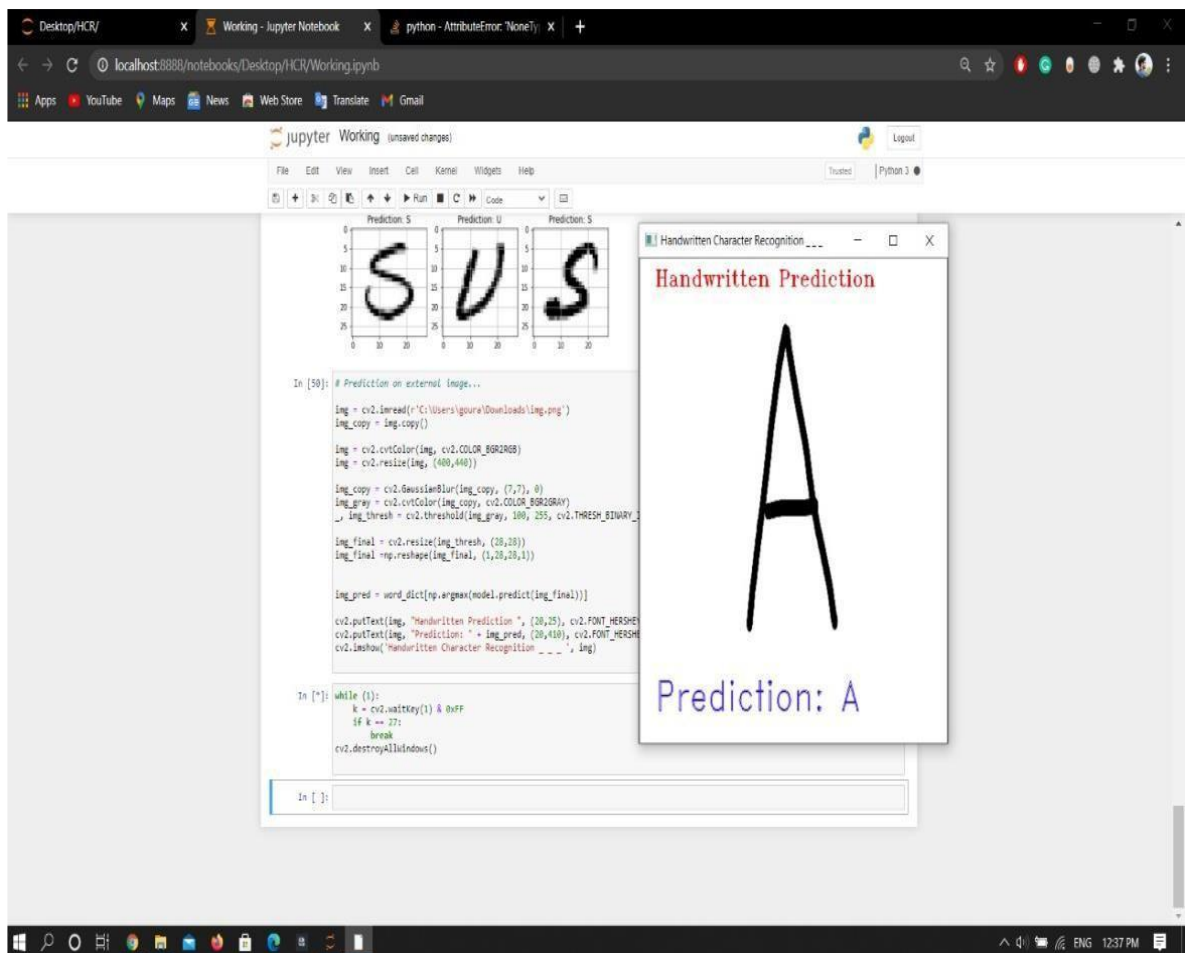


Fig.4.6 Prediction on External Image 4

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION :

Many regional languages throughout world have different writing styles which can be recognized with HCR systems using proper algorithm and strategies. We have learning for recognition of English characters. It has been found that recognition of handwritten character becomes difficult due to presence of odd characters or similarity in shapes for multiple characters. Scanned image is pre-processed to get a cleaned image and the characters are isolated into individual characters. Preprocessing work is done in which normalization, filtration is performed using processing steps which produce noise free and clean output. Managing our evolution algorithm with proper training, evaluation other step wise process will lead to successful output of system with better efficiency. Use of some statistical features and geometric features through neural network will provided better recognition result of English characters. This work will be helpful to the researchers for the work towards other script.

5.2 FUTURE WORK :

This work further extended to the character recognition for other languages. It can be used to convert the fax and news papers into text format. In order to recognize words, sentences or paragraphs we can use multiple ANN for classification. It can be used in post office for reading postal address.

REFERENCES

- [1] Megha Agarwal, Shalika, Vinam Tomar, Priyanka Gupta, "Alphabet Recognition using Neural Network and Tensor Flow", 2019. Volume-8, Issue-6S4, IJITEE, April 2019.
- [2] Singh, Sameer, Mark Hewitt, "Cursive Digit And Character Recognition on Cedar Database", Pattern Recognition, 2000. Proceedings. 15th international conference on. Vol. 2. IEEE 2000.
- [3] K. Gaurav and Bhatia P. K., "Analytical Review of Preprocessing Techniques for Offline Handwritten Character Recognition", 2nd International Conference on Emerging Trends in Engineering & Management, ICETEM, 2013.
- [4] Salvador España-Boquera, Maria J. C. B., Jorge G. M. and Francisco Z. M., "Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 4, April 2011.

APPENDIX

SOURCE CODE

```
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


# Read the data...

data = pd.read_csv(r"A_Z Handwritten Data.csv").astype('float32')


# Split data the X - Our data , and y - the predict label
```

```
X = data.drop('0',axis = 1)
```

```
y = data['0']
```

```
# 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)
```

```
# 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'}
```

```
# Plotting the number of alphabets in the dataset...
```

```
#viewing the dataset content in graph
```

```
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()
```

```
#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()
```

#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)

print("New shape of train data: ", test_X.shape)
```

```
# 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)
```

```
# CNN model...
```

```
model = Sequential()
```

```
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
input_shape=(28,28,1)))
```

```
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
```

```
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding =  
'same'))
```

```
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
```

```
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding =  
'valid'))
```

```
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', metrics=['accuracy'])  
  
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1,  
min_lr=0.0001)  
  
early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2,  
verbose=0, mode='auto')
```

```
history = model.fit(train_X, train_yOHE, epochs=1, callbacks=[reduce_lr,  
early_stop], validation_data = (test_X,test_yOHE))
```

```
model.summary()
```

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

```
# 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'])
```

```
#Making model predictions...
```

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

```
print(test_X.shape)
```

```
# 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'images\image1.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, "Handwritten Prediction", (20,25),  
cv2.FONT_HERSHEY_TRIPLEX, 0.7, color = (0,0,230))
```

```
cv2.putText(img, "Prediction: " + img_pred, (20,410),  
cv2.FONT_HERSHEY_DUPLEX, 1.3, color = (255,0,30))
```

```
cv2.imshow('Handwritten Character Recognition _ _ _', img)
```

```
while (1):
```

```
    k = cv2.waitKey(1) & 0xFF
```

```
    if k == 27:
```

```
        break
```

```
cv2.destroyAllWindows()
```

