Hand Written Digits Recognition using CNN

Machine Learning course, 3rd project

1st Ammar Farooq Khan
Department of Computer Science
Namal Institute Mianwali
Punjab, Pakistan
1602020@namal.edu.pk

2nd Muhammad Hamza
Department of Computer Science)
Namal Institute Mianwali
Punjab, Pakistan
1602028@namal.edu.pk

3rd Waseem Haider
Department of Electrical Engineering)
Namal Institute Mianwali
Punjab, Pakistan
1601022@namal.edu.pk

CONTENTS

***Abstract*—.**

In computer vision and machine learning applications, handwritten digit recognition is the most basic and important task. It leads the developers to further complex and big problems of recognition. There are lot of applications of digit recognition used in various domains like getting the serial number from number plates of different vehicles on government level, autonomous cars use same technology to take intelligent decisions, data entry for business documents, bank check processing and many more in almost every field. The ultimate purpose of our solution is to extend MNIST standard implementation to develop following feature. After gathering our own data, test the MNIST implementation on our hand written data which is an image containing an integer of up to 5 digits. Our program should be able to convert that image into equivalent number like 45321, 1243, 2, 13 etc. We are using CNN algorithm for this task because it is a most suitable for this problem. The 83 percent percent accuracy has been obtained from CNN algorithm.

I. INTRODUCTION

Now a days, it is a convenient method to show and transmit the information in the form of images. But there is a challenging part in it is that handwriting varies from person to person. A digit may be different in size, style, orientation and shape etc from the same number by different person. These variations make this task challenging. For large scale of applications, Image recognition is an important research area. For human beings, it is easy to read the printed or written documents, but using OCR (optical character recognition) this ability can also be indulged in machine as well.

OCR [4] is commonly used technology in which text is recognised in images like as photos and scanned documents. Basic purpose of this technology is to convert any type of image either it is handwritten, printed or in typed into machine readable text form. In early 1990s, OCR technology became popular to computerise the historic newspapers. After that, lot of improvement have been done in this domain. But nowadays, accuracy is much important than any other thing.

We can classified OCR into two categories based on acquisition of documents and text type [3]. If we look into text type, OCR is further divided into two types. One is HCR (Handwritten character recognition) while second is PCR (Printed character recognition).

In real world, hand written recognition of characters is very useful. Several methodologies have been utilized in handwriting recognition field like neural networks, Structural methods, syntactic methods and statistical methods [3].

II. BACKGROUND

In this section, we are giving an overview of Neural Networks and CNN in detail.

A. Neural Networks

Neural Networks commonly name used as Artificially neural networks are not very new technology. First idea about them was introduced in 1944 by Warren McCullough and Walter Pitts, who were researchers in University of Chicago at that time [8].

In neural networks, computer by analyzing the training dataset learns to perform desired tasks. The training dataset is labeled that's why this falls under the category of supervised machine

learning algorithms. In ANN, for example we give thousands different types of images of cars to model to predict the actual car which we want to recognise, by extracting the visual patterns in given pictures to correlate.
NN are proposed to copy human brain. Neural net comprises of millions of small processing nodes working jointly. Today's neural nets are multi layered. One layer feed forward its output into next layer as input. They are unidirectional. One node can get inputs from many nodes of previous layer. The term weight is also associated with nodes. It means that when node will get input from different nodes, it has to give priority to some nodes who are important for it. Those nodes will get high weights as compared to others nodes. There is threshold concept also here, in which node only pass output when it is receiving data accumulated above our set threshold otherwise it will not contribute to next nodes. Below is a diagram [1] of Neural Network showing the different layers, how inputs and outputs are parsing.

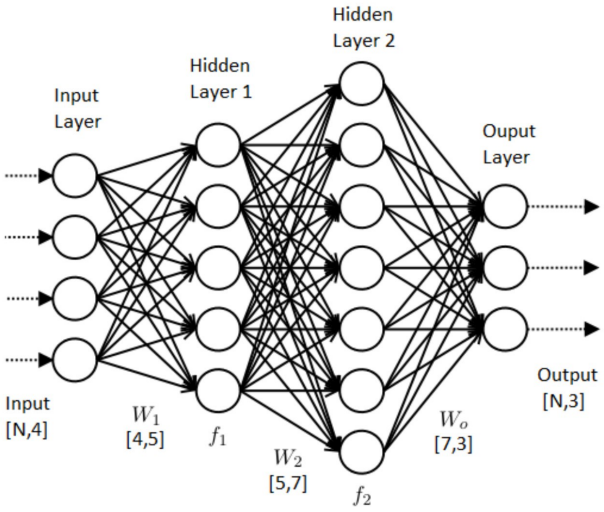


Fig. 1. [1]Neural Network

In Neural Networks, this network is passed with initially random weights and thresholds as initialized above. The first trainable neural network, the Perceptron, was demonstrated by a psychologist Frank Rosenblatt in 1957 at the Cornell University. The Perceptron's design was also like of the modern neural net, except that with adjustable weights and thresholds it had only one layer, between input and output layers. There are different types of neural network which are only described in the form of images.

Feed Forward Neural Network (FFNN)
Deep Feed Forward Neural Network (DFF)
Recurrent Neural Network (RNN)
Extreme Learning Machine (ELM)
Deep Convolutional Network (DCN)
Convolutional Neural Network (CNN)

.

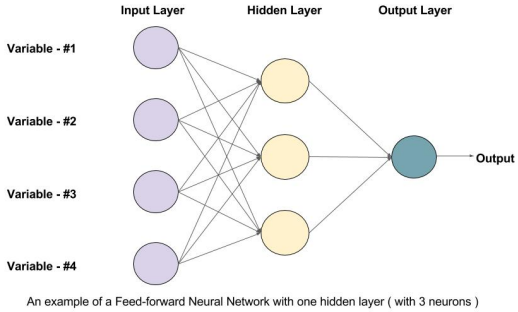


Fig. 2. [5]Feed Forward Neural Network

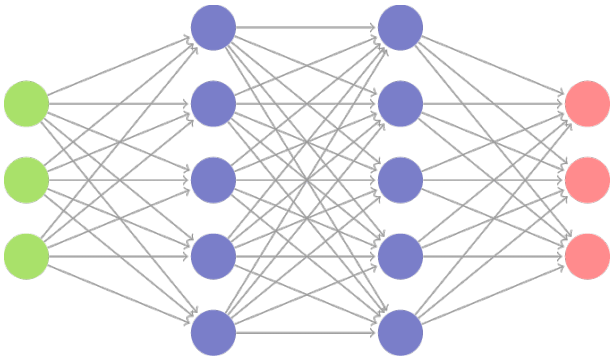


Fig. 3. [6]Deep Feed Forward Neural Network

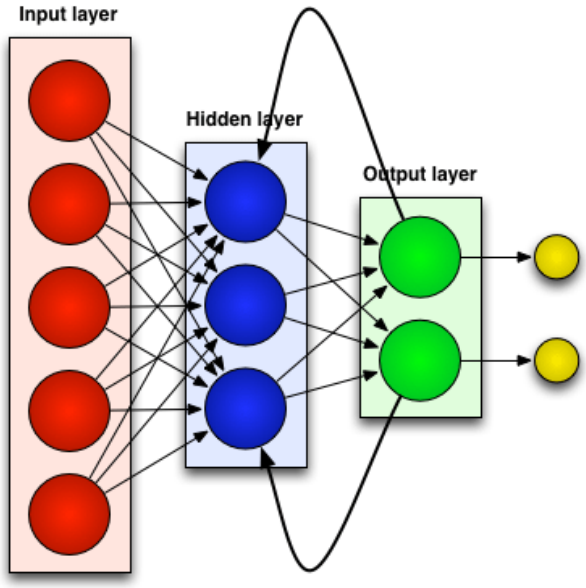


Fig. 4. [7]Recurrent Neural Network

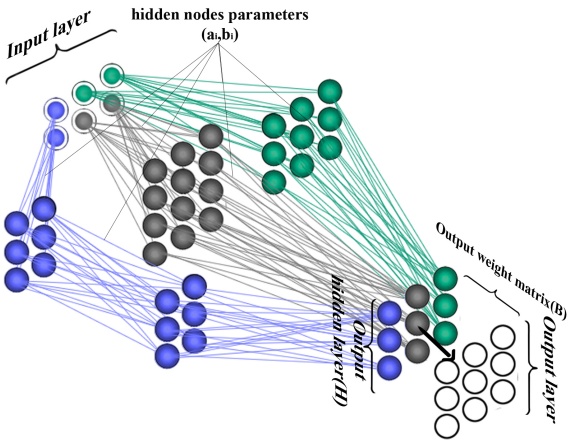


Fig. 5. [12]Extreme Learning Machine

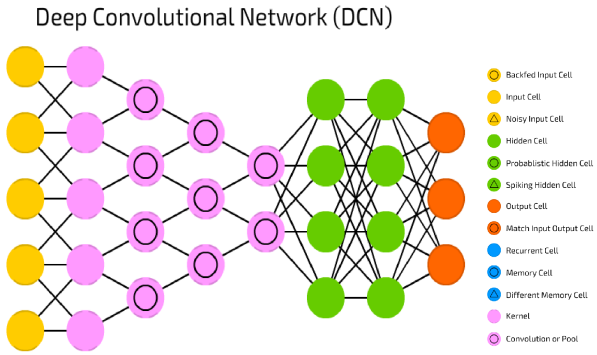


Fig. 6. [9]Deep Convolutional Network

B. CNN

Now in this part, we are trying to explain the basic concepts of CNN with respect to digit classification.

Why Convolution layer is used? [10]
When concept of CNN was not presented until 1998 by Yann LeCun for digit classifications, people used different methods like logistic regression, knn, support vector machine etc to classify images. In those parts, pixel values i.e 28*28 were considered as a feature and in that case there was 784 features. But in this method, we lose between pixels a lot of spatial interaction. In opposite CNN gives better solution in a way that it extract information by down-sample the image from adjacent pixels by convolution into features and after that it uses prediction layer to predict the testing dataset.

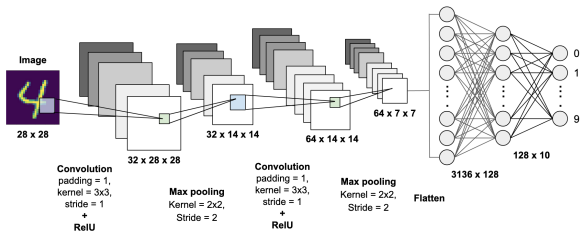


Fig. 7. [9]Convolutional Neural Network

How Convolution layer work? [10]
We use multiple convolution filters that run on image to get a dot product. Different features are extracted from the image by each filter.
Consider a filter having size of 3*3 and image size is 5*5. We element wise multiply image values who match the size of kernal and then after adding get a single value for that feature cell.

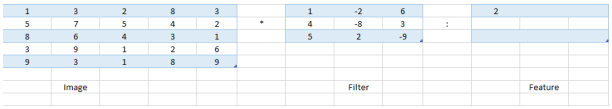


Fig. 8. Convolutional operation 1

1*1+3*(-2)+2*6+5*4+7*(-8)+5*3+8*5+6*2+4*-9 = 2
After multiplying, we get the following result.

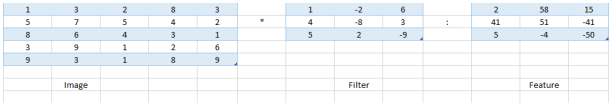


Fig. 9. Convolutional operations

In above computation, we slide the kernel by 1 pixel. This is known as **stride**. We can shift stride by 2 also. Then our feature matrices will be 2*2. If we want the feature, same size of image size then we have to introduce the concept of padding **Padding** is a technique in which we simply add zeros around the image to increase it dimension.
After this, we do max pooling. **Max Pooling** also reduce the size of matrix of convolved features. Given below is an example which shows layer of max pooling having size of 2 with stride of 1.

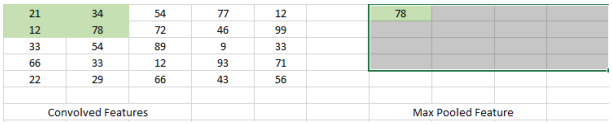


Fig. 10. Max Pooling step 1

Now the whole metrics of max pooled is given below.

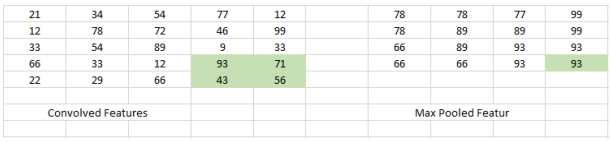


Fig. 11. Max Pooling

Max poling is not the only type of pooling. There are also average pooling and min pooling in field.

III. Dataset

A. MNIST dataset

The MNIST (Modified National Institute of Standards and Technology database) is a large database of hand written digits which is normally used for different image recognition systems. The data is divided into two parts, training and testing which is used in the field of machine learning oftenly. These images are also further normalized to fit into 28*28 pixels and centered in a fixed-size image. This is the best data for those who just want to learn techniques and methods of pattern recognition on actual data without spending more efforts on formatting and preprocessing.
The training images are 60,000 while testing images are 10,000 in MNIST dataset. This is small chunk of large dataset, which is available on NIST. [13] Given below is a sample of 9 images from MNIST dataset.

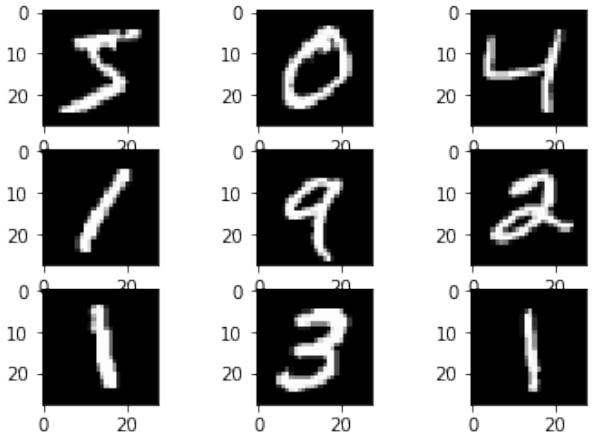


Fig. 12. MNIST Dataset

B. Own Written Dataset

In our own gathered dataset, we have 100 images of 0 to 9 digits. Which means that every digit has 10 images. We'll give these images to MNIST standard implementation to cross check the results. Below in figure a sample given in non processed form.

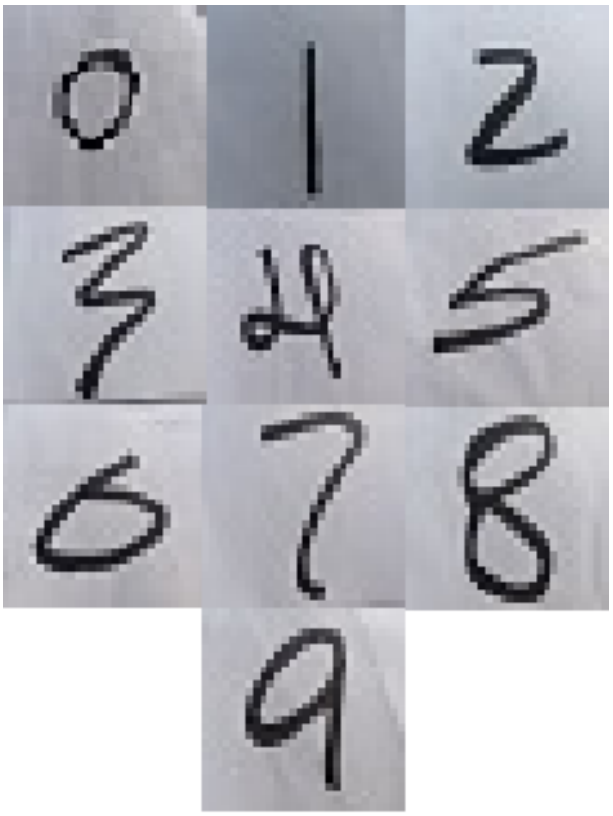


Fig. 13. OWN Dataset containing 1 digit

After that we have 20,20 images for all, two to five digits. We modified the existing simple implementation to get correct recognition of more than 1 digit in an image. Below are a samples given in non processed form of 2 digits, 3, 4 and 5 digits.

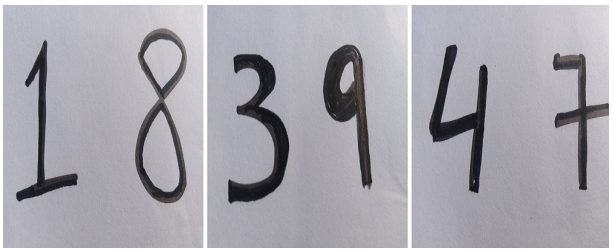


Fig. 14. 2 digits OWN Dataset before processing

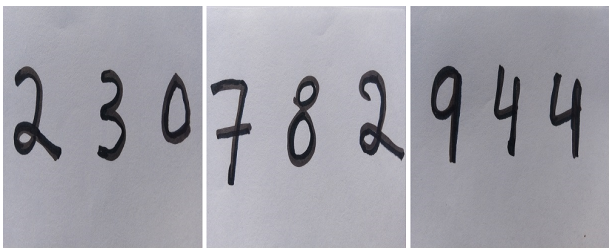


Fig. 15. 3 digits OWN Dataset before processing

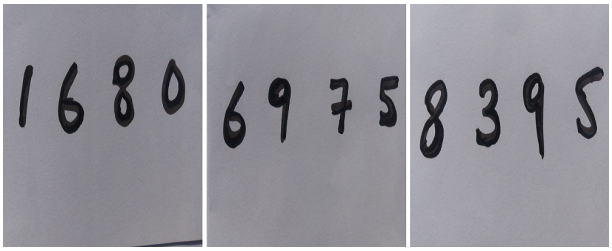


Fig. 16. 4 digits OWN Dataset before processing

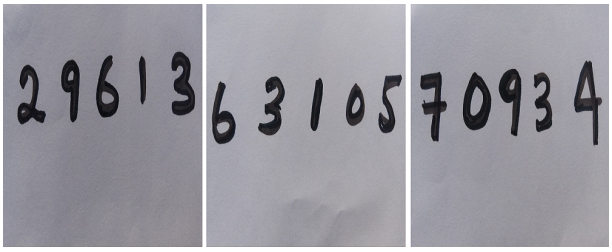


Fig. 17. 5 digits OWN Dataset before processing

C. Challenges in dataset

We faced lot of problems in our dataset because we wrote our dataset on paper. There were noise issues, doing so as compared to those who made dataset on different softwares. Their background is clear white, so digit can easily be identified. When we gave our dataset to model to predict. That dataset was written firstly by ball pen which was too thin. When we resize it into 28*28, then it was not recognisable. Then we again wrote the dataset with marker with 3x thickness. Then our model was able to recognise. Another challenge is detection and localizing the multiple digits from one image. We used cv2 contour() method to detect the edges of each digit from an image. But then we faced the new challenge of order of digits from left to right. Contour() method mix the digits during picking the coordinates of different digits. For example an image consist of 3072, some times contour localize correct 3072 or sometimes 2307, 0723 etc. It was the most time consuming challenge that we faced and solved this. We solve this using sorting algorithm. Contour() return (y, x, width, height) points/coordinates for one digit localizing or one bounding box. We sort these coordinates on the based of column values, bounding box coordinates are according to our need.

IV. Methodology

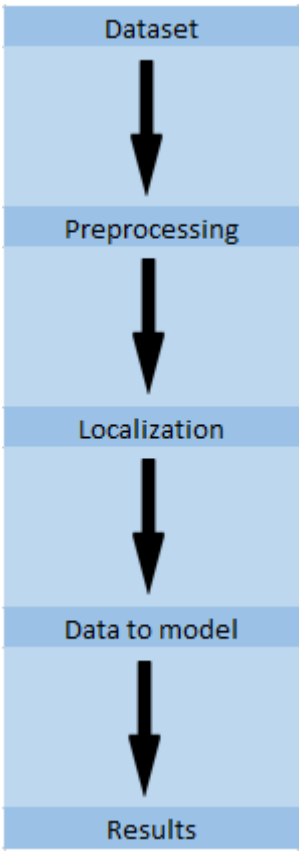


Fig. 18. Dataflow

As shown in Fig no 18, we are doing basically three steps, preprocessing first, localization of data and after this, we are giving refined data to CNN model which is showing results. All these steps are written below after code explanation in detail.

First of all code is elaborated below [2], we load the dataset as in given below.

```
# load train and test dataset
def load_dataset():
  # load dataset
  (trainX, trainY), (testX, testY) = mnist.load_data()
  # reshape dataset to have a single channel
  trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
  testX = testX.reshape((testX.shape[0], 28, 28, 1))
  # one hot encode target values
  trainY = to_categorical(trainY)
  testY = to_categorical(testY)
  return trainX, trainY, testX, testY
```

Fig. 19. Loading the dataset

After loading, we normalized the pixel values of gray scale images to 0-1.

In CNN sequential model, all layers used RELU activation model. Given below is code image to show the steps.

```
# define cnn model
def define_model():
  model = Sequential()
  model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(28, 28, 1)))
  model.add(MaxPooling2D((2, 2)))
  model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'))
  model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'))
  model.add(MaxPooling2D((2, 2)))
  model.add(Flatten())
  model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  # compile model
  opt = SGD(lr=0.01, momentum=0.9)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

Fig. 20. CNN model implementation

When we test the MNIST testing data, it gave us 99.180 percent accuracy as given below after 10 epoch.

```
Testing on MNIST test data:
> 99.180
```

Fig. 21. results on MNIST

A. Preprocessing

For achieving best results, preprocessing is the most important part in machine learning algorithms in which data is refined into proper form. For that, we used 3 different techniques, re scale data, binarize data, and in last standardized data [11].

1) Rescale data: In this part, first of all we got a raw image. We converted that high definition rgb images into gray scale because it is easy to processed as compared to rgb image. In gray scale, either it will be white or black which is easy to recognise. So conversion to gray scale is important to further proceed.
We have a gray scaled image, but we have to resize it now. Because in CNN, images may be differ in resolution, so we have to resize it to all have same scale to pass. Ideally we have to give a 28*28 resolution image to continue.

2) Binarize data: In Noise removal part, we set if image has a pixel value more than 150 then it should considered as 1, while if pixel value is less than 150 it is considered as 0 as earlier discussed in normalizing the pixel part. This is called threshold our data or binarizing the data.

3) Standardization: Standardization is valuable method to change properties with a Gaussian distribution and differing means and standard deviations to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.
We can standardize data with the StandardScaler class using sklearn.

B. Localization

This process is typically used to locate objects and identified the boundaries of each digit. Evaluating the location coordinates obtained from the model, for the object in the image. In our project, we have more than 1 digit in majority testing images. So our segmentation part first locates a different kind of digits as an object and then separates every digit from one another by drawing boundaries using the X-axis and Y-axis coordinates. For bounding the box, we used contour function. This function bound our all digits easily. Contour has many features but we used the bounding rectangles. And in it, we used Straight Bounding Rectangle which bounds the whole image straight. This don't try to minimize the area of rectangle.

```
#finding bounding rectangle
rects = [cv2.boundingRect(con) for con in cntr]
```

Fig. 22. Contour function

Given below are the images in which it is localizing the 1 digit in every image.

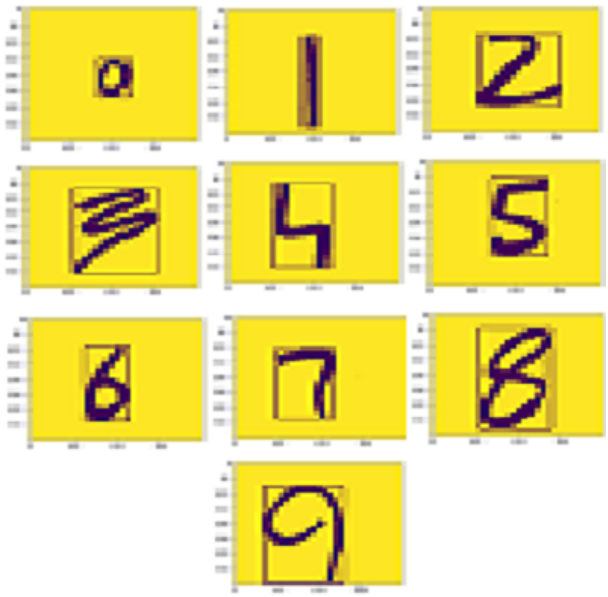
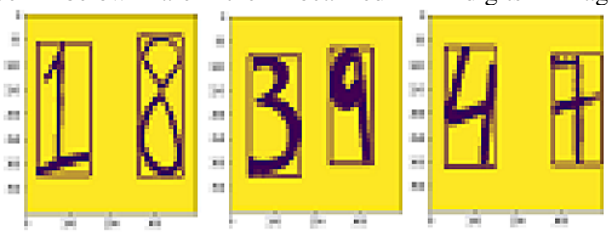


Fig. 23. digits with localization

Given below are the localized 2 digits images.



Given below are the localized 3 digits images.

Fig. 24. 2 digits with localization

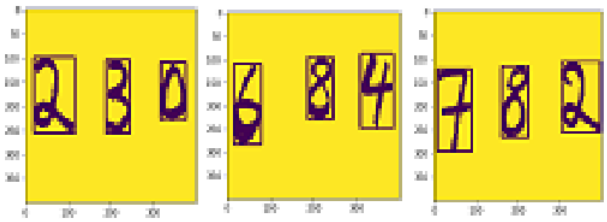


Fig. 25. 3 digits with localization

Given below are the localized 4 digits images.

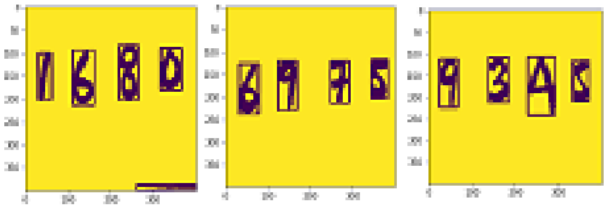


Fig. 26. 4 digits with localization

Given below are the localized 5 digits images.

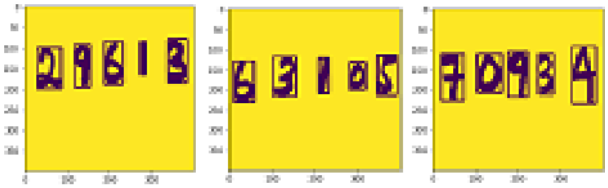


Fig. 27. 5 digits with localization

C. Parsing the refined data to Model

We gave the refined data to CNN model to predict. CNN model is already explained.

D. Results

In this part, results are presented in written form or the form of graphs.
First, We test trained model on own single digits dataset of 100 images. Images were 0 to 9. Accuracy was 83 percent on our 1 digit dataset.

Images	Total	Correct Predictions	Bad Predictions	Accuracy
Zero	10	7	3	70 %
One	10	9	1	90 %
Two	10	10	0	100 %
Three	10	9	1	90 %
Four	10	8	2	80 %
Five	10	9	1	90 %
Six	10	6	4	60 %
Seven	10	9	1	90 %
Eight	10	8	2	80 %
Nine	10	8	2	80 %
Total	100	83	17	83 %

Fig. 28. Result of 1 digit

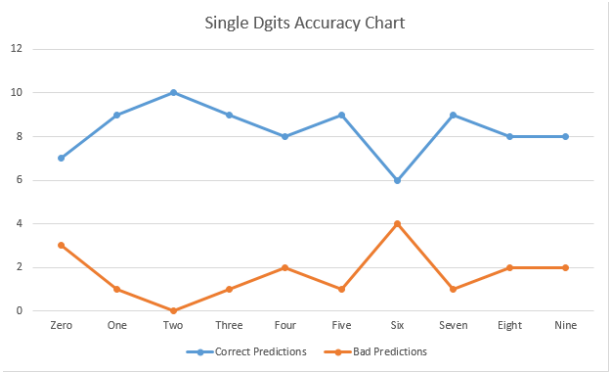


Fig. 29. Result of 1 digit in graph

Same in the other data set of 2,3,4 and 5 digits; accuracy is shown below.

Images	Total Images	Correct Predictions	Bad Predictions	Accuracy
Two Digits	20	17	3	85 %
Three Digits	20	16	4	80 %
Four Digits	20	17	2	85 %
Five Digits	20	13	7	65 %

Fig. 30. Result of 2,3,4 and 5 digits

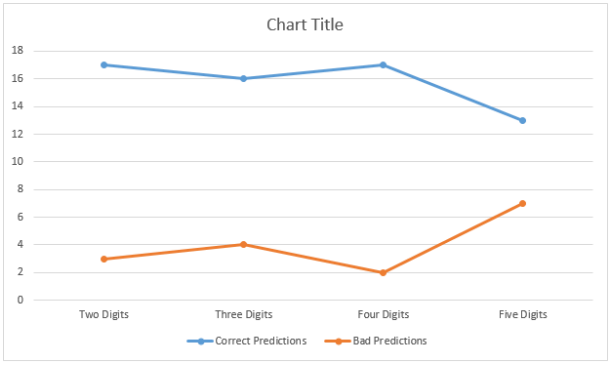


Fig. 31. Result of 2,3,4 and 5 digits in graph

V. Conclusion

In this work, We predicted the digits extracted from images of dataset from MNIST as well as own dataset. We have a model from machinelearnigmastery then first MNIST labeled

dataset was given to it. Model correctly identified with an accuracy of 99 percent at MNIST dataset. Further we made our challenging dataset ranging from 1 digit to 5 digits. Then we first gave the 10,10 images of each 0 to 9 digits to model for prediction, it gave us 83 percent accuracy. The accuracy in 2 digits images is 85 percent, 80 percent in 3 digits, 85 percent in 4 digits and at last 65 percent accuracy in 5 digits images. This task is highly challenging and informative.

REFERENCES

[1] Jayesh Bapu Ahire. *The Artificial Neural Networks*. Aug 24, 2018.

[2] Jason Brownlee. *How to Develop a CNN for MNIST Handwritten Digit Classification*. machinelearningmastery.com, May 8, 2019.

[3] Hruday M Likitha N Girish B G Deepak Gowda D A, Harusha S. *Implementation of Handwritten Character Recognition using Neural Network*. IJIRST –International Journal for Innovative Research in Science Technology— Volume 5, 2019.

[4] docparser. *What Is OCR And What Is It Used For?*

[5] Vikas Gupta. *Understanding Feedforward Neural Networks*. October 9, 2017.

[6] Andreas Holzinger. *Deep Feedforward Neural Network*. Oct 2017.

[7] JROELL. *Understanding Recurrent Neural Networks: The Preferred Neural Network for Time-Series Data*. January 26, 2017.

[8] MIT news office Larry Hardesty. *Explained: Neural networks Ballyhooed artificial-intelligence technique known as "deep learning" revives 70-year-old idea.* MIT News Office • Building 11-400 Massachusetts Institute of Technology • Cambridge, MA 02139-4307, April 14, 2017.

[9] Dominik Mielczarek. *Deep Convolutional Network*.

[10] Krut Patel. *Convolutional Neural Networks — A Beginner's Guide*. towardsdatascience.com, Sep 8, 2019.

[11] Abhishek Sharma. *Data Preprocessing for Machine learning in Python*. geeksforgeeks.org.

[12] BERGHOUT Tarek. *multilayers perceptron based Extreme Learning Machine*. 2019.

[13] NYU; Corinna Cortes Google Labs New York; Christopher J.C. Burges Microsoft Research Redmond. Yann LeCun, Courant Institute. *THE MNIST DATABASE of handwritten digits*.