International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/IJSRCSEIT

Handwritten Digit Recognition System

Shubham Mendapara*, Krish Pabani, Yash Paneliya
B.Tech. Scholar, DEPSTAR (CSE), Charotar Science University of and Technology, Gujarat, India

Article Info
Volume 7, Issue 5
Page Number: 76-85

Publication Issue :
September-October-2021

Article History
Accepted : 07 Oct 2021
Published : 14 Oct 2021

ABSTRACT
Recently, handwritten digit recognition has become impressively significant with the escalation of the Artificial Neural Networks (ANN). Apart from this, deep learning has brought a major turnaround in machine learning, which was the main reason it attracted many researchers. We can use it in many applications. The main aim of this article is to use the neural network approach for recognizing handwritten digits. The Convolution Neural Network has become the center of all deep learning strategies. Optical character recognition (OCR) is a part of image processing that leads to excerpting text from images. Recognizing handwritten digits is part of OCR. Recognizing the numbers is an important and remarkable subject. In this way, since the handwritten digits are not of same size, thickness, position, various difficulties are faced in determining the problem of recognizing handwritten digits. The unlikeness and structure of the compositional styles of many entities further influences the example and presence of the numbers. This is the strategy for perceiving and organizing the written characters. Its applications are such as programmed bank checks, health, post offices, for education, etc.

In this article, to evaluate CNN's performance, we used the MNIST dataset, which contains 60,000 images of handwritten digits. Achieves 98.85% accuracy for handwritten digit. And where 10% of the total images were used to test the data set.
Keywords: Handwritten digit recognition, Convolution Neural Networks (CNN), MNIST dataset, TensorFlow, Keras, OpenCV, Deep Learning

I. INTRODUCTION

Image are easily the human brain easily processes and analyzes images. When we see a particular image, the brain can easily divide it in chunks and recognizes its different parts. The brain goes through this step naturally, which includes analyzing these images as well as comparing and differentiating their multiple features and characteristics from what the brain already recognized in order to recognize these characteristics. Image Processing is a branch of computer science that attempts to do the identical thing with computers.

The field of image processing entails analyzing images to take out some useful and crucial information from images. Image processing converts images into a digital form(matrices) which are machine readable. It executes categorical algorithms on them and displays the results in a higher-resolution image or with any of their characteristics and features that can be used to extract high-priority information. Optical Character Recognition (OCR) is a sub-field of image processing that deals with the process of recognizing characters from an image (OCR). This approach examines an image containing scanned text or handwritten characters and attempts to recall them using a variety of algorithms. The primary applications of OCR are recognized handwritten characters. In this article, we'll look at how to create a machine that can identify handwritten digits. First, we'll read images of handwritten digits from the MNIST dataset, which is extremely well arranged, and try to figure out which digit each image represents. For this, we will be using artificial neural networks (i.e., CNN) which is considered to be the best for the purpose. Artificial neural networks are similar to the neural network of the human brain which is a set of I/O units that are linked together (i.e., neurons) where each connection has a weight associated with it, and each relation has a weight associated with it. We may choose to use ANN to build statistical models from massive datasets. This paradigm assists us in comprehending image classification. A neural network is made up of neurons (also called nodes). These nodes are connected in some way. Then each node has a number, and each connection has a weight. These neurons are split between three layers distinguished as the input, hidden and output layer. Input layer receives inputs and passes it to other layers. The number of input neurons is approximately the same as the number of functions. Hidden layer applies various convolutions and transformations to the inputs before passing them further. The weights are updated as the network is conditioned to be superfluous predictive and precise. Small random numbers, such as 0 to 1, are often used to initialize neuron weights. The output neuron is the predicted feature in the output layer.

II. LITERATURE REVIEW

As we know there is no computer which can beat the level of the human brain. So due to these inefficiencies in computers we use artificial neural networks to make them somehow efficient like humans. Human brain easily processes and analyzes images. Brain automatically identifies and recognizes the elements and features of images. Image processing is a field which deals with enabling machines to do such tasks that our brain can do with images.

Nowadays we see that technology is increasing repeatedly and many options are available to perform Handwritten digit recognition. But CNN plays a very crucial role in many image processing applications. CNN is used for detection of data loss (fault) and accuracy of the application. In paper we have shown use of deep learning with others such as CNN using TensorFlow, Keras, OpenCV. These algorithms are used widely by researchers as experiments for theories of machine learning. Many researchers are using this technique other than machine learning algorithms such as SVM, KNN, and RFC etc., they prefer to use CNN because it gives high accuracy in image classification, video analysis etc. Moreover, it is also used in sentiment recognition, researchers are going for more accurate modal and less error correction [2].

The CNN (Convolutional Neural Network) has brought a revolutionary change in the field of machine learning. Particularly in character recognition. In 2003, Simard et al. introduced a general CNN architecture for analysis of visual documents and the sophisticated training method of neural networks [1]. Mahmoud M. Abu Ghosh compared CNN, DNN, DBN strategies for making understanding which neural network is useful in the computer vision field especially in image processing like OCR. According to their findings, DNN gives accuracy of around 98.85% but fails against CNN approach in team of time [3]. Badri Narayanan et al. performed a convolution neural network for semantic segmentation. An encoder network, a decode network, and a pixel-wise classification layer make up the segmentation. For decoding, the proposed approach used max-pooling indices, and the result was quickly observed as good results. This method was also analyzed with past existing techniques for more deep understanding [4][6].

III. METHODOLOGY AND CLASSIFICATION

For developing the deep learning model, we will be using Convolutional Neural Network. Convolutional networks are ingeniously useful in biological psychology since the sequence of connections between neurons mimicked the animal visual cortex's association. In a limited area of the chromatic field known as the receptive field, independent cortical neurons respond to solitary stimuli. Contrary neurons' receptive fields partly overlap, allowing them to occupy the whole visual field. One of the main propositions of OCR is recognizing handwritten digit characters. In this instance, we will emphasize the structure of a mechanism by considering recognizing handwritten digits. We'll read a variety of images containing handwritten digits extracted from the MNIST database and try to figure out which digit is represented by each image. MNIST data is illustrated in the IDX file format.

For that we will be using artificial neural networks (i.e., CNN) which is considered to be the best for the purpose of taking advantage of it for using it in a handwritten digit recognition system to create models. CNN extracts the feature maps from the 2D image. This CNN predicts the mapping of image pixels with near space satisfactorily than having a fully connected layer of neurons. This machine learning model is used to recognize people's handwriting digits. CNN resembles the behavior of human/animal brain neurons, that's why it will be the best algorithm to go with. This task is divided into different phases.

IV. DATASET

We have taken the dataset from the MNIST dataset which contains 60,000 images of handwritten digits (see Figure 1, zero through nine), all of which are grouped into one file. Each of the images is gray scaled and positioned in a fixed size of 28 * 28 pixels. It represents a number. It forms an array that can be flattened to a dimension vector 28 * 28 = 784. We found that there was no pattern or order in how the images were organized in the file. Digital images are represented in the form of multidimensional matrices, the cells of matrices represent the pixels. In addition, each picture has a label indicating the number shown. In addition, the dataset contained no noise or major problems to be processed, so we were able to use it without any preprocessing operations. These images are not directly visible. To display these images, we have to use matplotlib. This dataset contains handwritten numbers of around 250 people, 50% of them were Census Bureau employees and the remaining were students [7].

The system that will be built will be evaluated using pictures of handwritten numbers to assess its performance and accuracy, as well as to determine its success rate. As for the result, putting together a test set is critical. The test set contains an example of the pictures that must be compared to the images in the reference set to identify the handwritten digits. This record was formed using the MNIST dataset. There were 60,000 pictures in the original file, each with a distinct number. We divided the dataset into 9:1 ratio for training and testing. So, we split it into two parts of 54000 and 6000 frames. These 6000 images are used to test the model. MNIST data is illustrated in the IDX file format as shown in Figure 1.

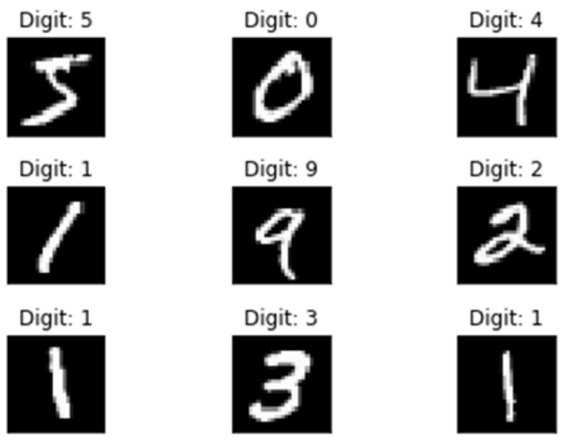


Figure 1. Sample images of MNIST handwritten digit dataset

V. MODEL ARCHITECTURE

Convolutional Neural Network is a special type of multi-layered neural network which is modeled to identify visual patterns from pixel images with nominal preprocessing. The basic CNN includes four components, like the convolution layer, the pooling layer, the flatten layer, the output layer and the image is passed through all these layers. The image at the end reaches the output layer. Pooling layer is optional depending on the image type whether is required or not. CNN is formed by various layers of neurons that transfer input parameters to the output. To choose which neuron to activate in each layer there is an activation function such as ReLU, Sigmoid, SoftMax etc. ReLU is a widely used activation function. After the dataset is loaded, we separated data into two parts X and Y in the following way. As we know the original file contained 60,000 images representing different digits. For training (x) and testing(y) we will split the dataset into 9:1 ratio. So, we split it into two parts of 54000 and 6000 images.

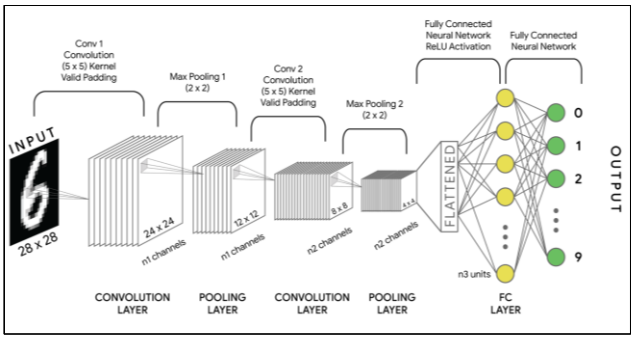
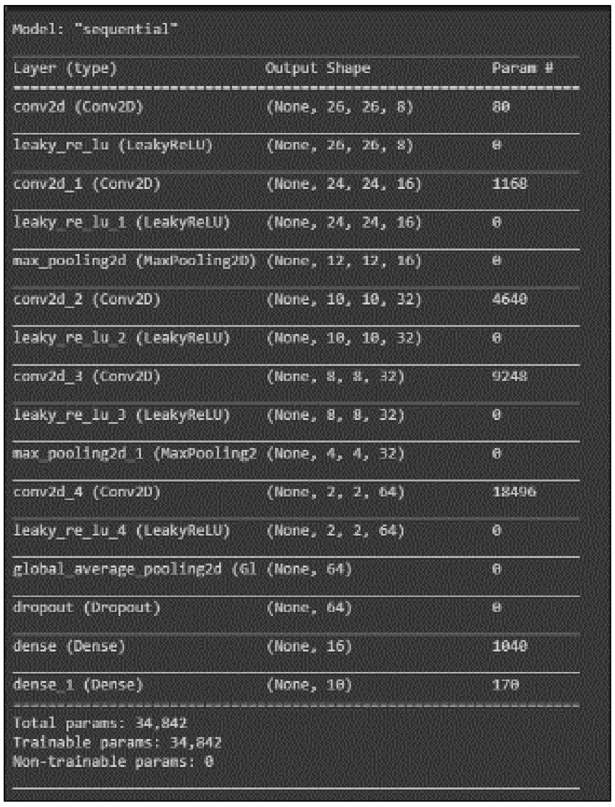


Figure 2. Model Layer’s



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 8)         80
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 26, 26, 8)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 16)        1168
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 24, 24, 16)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 32)        4640
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 32)          9248
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 8, 8, 32)          0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 32)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 64)          18496
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 2, 2, 64)          0
_________________________________________________________________
global_average_pooling2d (Gl (None, 64)                0
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 16)                1040
_________________________________________________________________
dense_1 (Dense)              (None, 10)                170
=================================================================
Total params: 34,842
Trainable params: 34,842
Non-trainable params: 0
_________________________________________________________________
```

Figure 3. Model Summary

A. Convolution Layer

The first layer of CNN is a 2D convolution layer that folds the input image and prepares it for the desired input. It consists of a built-in hidden layer to customize the classification of handwritten images. The convolution layer filters the image with a smaller pixel filter (known as Kernel) to reduce the size of the image without losing the original relationship between the pixels. We will apply convolutions to the input image and the output will be provided to the next layer. This is equivalent to the response of a neuron in the visual layer to a specific input. Each neuron only processes information relevant to its manipulable region.

Parameters of Convolution Layer:

1. Kernel Size: 5 x 5
2. No. of filters: 8
3. Activation function: leakyReLU
4. Input shape: (28,28,1)

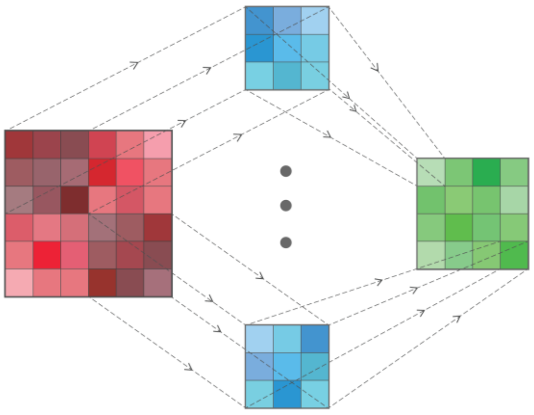


Figure 4. Visualization of Convolution operations

B. Pooling Layer

After performing convolutions on the images, feature maps are created which highlight the features of the input. However, this feature map contains very precise locations of features such as vertical lines, horizontal lines, and so on. Result of Maps with different characteristics is created due to minor differences in the position of the features in the image. We therefore have to down-compute the feature map received from the convolutional layer.

We use a pooling layer for resolving this problem. The pooling layer is similar to down sampling and signal processing. It approaches the characteristic map, so a small deviation cannot change the correct characteristic map. This results in a lower-resolution version of the input picture that retains the large or significant structural elements but excludes small information that might be irrelevant to our project.

For our model, we use a Max Pooling plane that extracts the maximum value of the submatrix of the specified size of the feature map received from the convolutional layer. From this level we get a summarized version of the feature map.

Parameters for Pooling layer:

1. Pool size: 2 x 2
2. Strides: 2 x 2

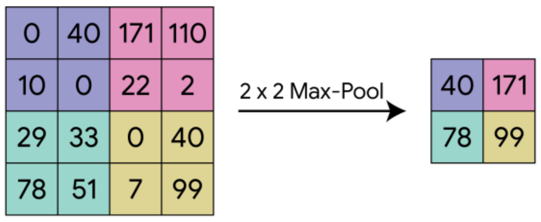


Figure 5. Visualization of Pooling operation

C. Second Convolutional Layer & Max Pooling Layer

After applying Max Pooling, we add another Convo level and another Max Pooling layer. Pooling that determines the highest or largest value for each patch on each feature map is known as peak pooling or maximum pooling. Since it uses maximum values from each sub-region and provides as much information as possible, the maximum pooling layer is generally beneficial in modern applications. This leads to greater generalization and quicker convergence.

Parameters for Convo. layer:

1. Kernel size: 5 x 5
2. No. of filters: 16
3. Activation function: leakyReLU
4. Strides: 1

Parameters for MaxPooling Layer:

1. Pool size: 5 x 5
2. Strides: 2 x 2

D. Flatten Layer

In the final stages of CNN, we have flattening and completely connected layers. In the flatten layer, the data is converted into a one-dimensional array for input of the next level. To build a single long function vector, we minimize the contribution of the convolutional layers. It's also linked to the totally connected layer, which is the final classification model. To put it in another way, we put all of the pixel data on a single line and link it to the last layer. And again.

E. Output Layer

This is the final layer of our model. This model will have 10 neurons. Each neuron will represent a number from 0-9. So, from this layer we'll get our predicted number. After multiple layers of mixing and pairing, we will need the output in the form of a section. Layers of convolution and pooling will only be able to remove the features and reduce the number of parameters from the original images.

However, to produce the final result we need to use a fully integrated layer to produce the result equal to the number of classes we need. It is difficult to reach that number with convolutional layers. Convolution layers generate 3D activation maps while requiring output whether the image is in a certain category or not. The output layer has a loss function as cross-entropy of the categories, calculating the error in prediction. When the pass is completed, the regression begins to regain weight and the bias of the error and to reduce the loss.

VI. SOFTWARE ARCHITECTURE

A. Algorithm Used

In figure 6, we have described the algorithm used for our proposed system. In the system there are two ways to provide the image as input. First is by drawing on canvas and the other is uploading a photo directly of any size. After providing the image or drawing on canvas, the image will be fed to our deep learning model through a REST API as a POST request. All the preprocessing tasks like gray scaling of image, image resizing, contour identification etc. are done in the API.

B. Image Preprocessing

Before we feed the received image into the model, we have to process the image in the required format. First, we convert the image to grayscale because feature extraction and identification is easy in grayscale. After that image is thresholded to remove unwanted noise from the image. Contours are detected in the thresholded image and the region of interest is resized to 28*28 pixels with a single-color channel. Then we normalize the pixel values to make computation easier by dividing them by 255. Now our image is ready to feed into the model. The prediction is made and the result is returned to the system and displayed.

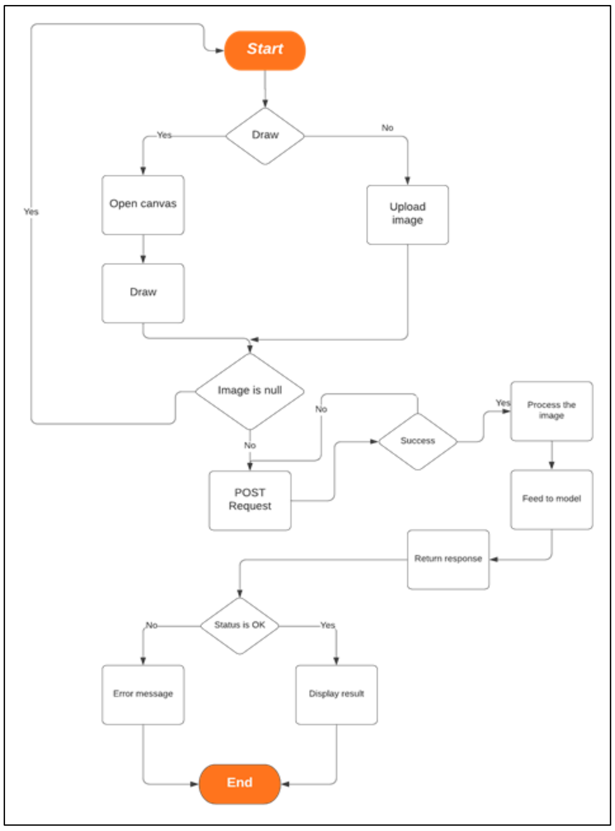


Figure 6. Flowchart of proposed system

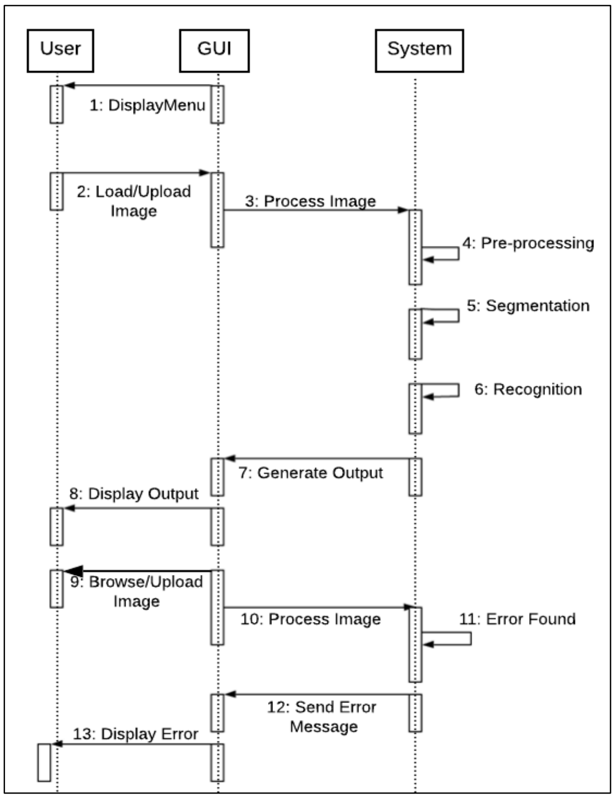
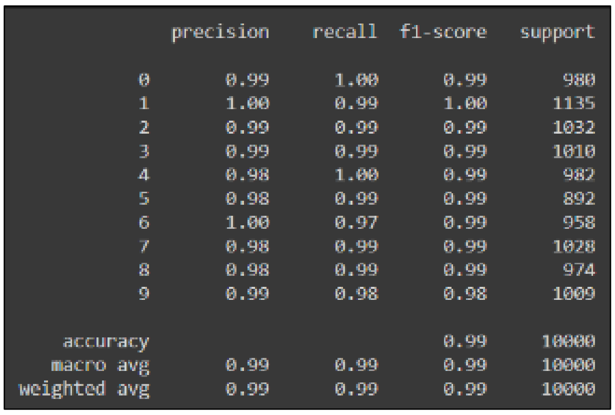


Figure 7. Sequence Diagram of Application

VII. RESULTS



```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       980
           1       1.00      0.99      1.00      1135
           2       0.99      0.99      0.99      1032
           3       0.99      0.99      0.99      1010
           4       0.98      1.00      0.99       982
           5       0.98      0.99      0.99       892
           6       1.00      0.97      0.99       958
           7       0.98      0.99      0.99      1028
           8       0.98      0.99      0.99       974
           9       0.99      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

Figure 8. Classification Report

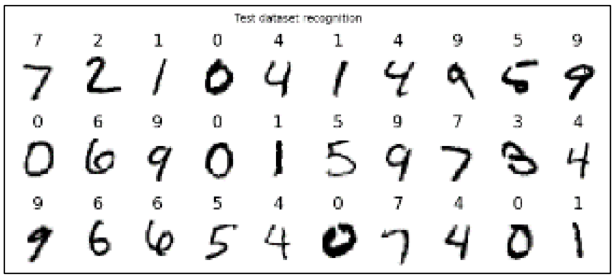


Figure 9. Test dataset recognition

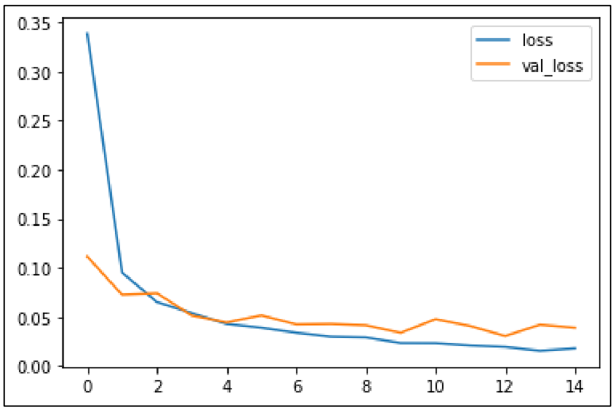


Figure 10. Training and Validation loss

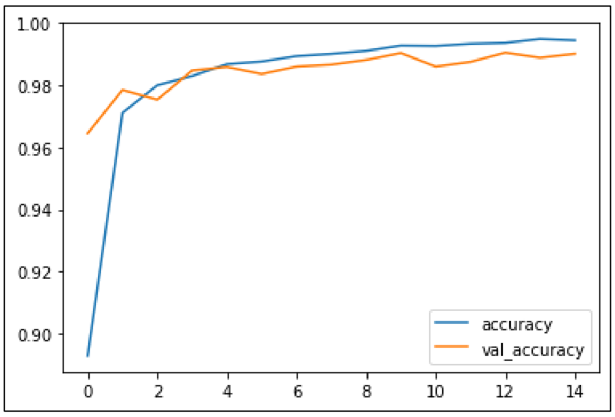


Figure 11: Training and Validation Accuracy

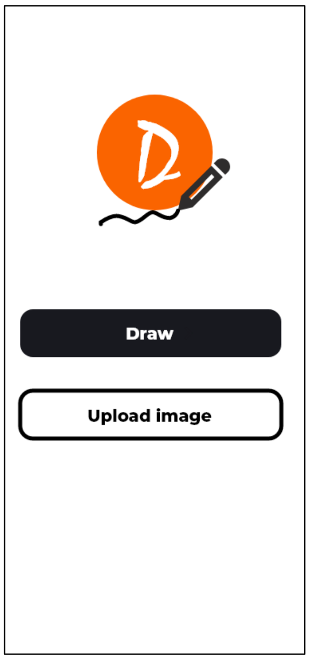


Figure 12. Proposed system for the aim

Users have two options for providing the input, by drawing on canvas or by uploading an image.

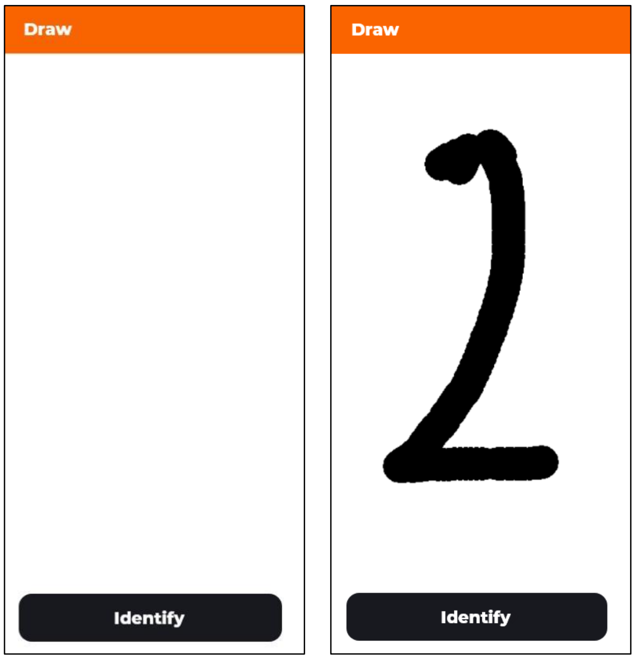


Figure 13. We are providing an input by drawing on canvas.

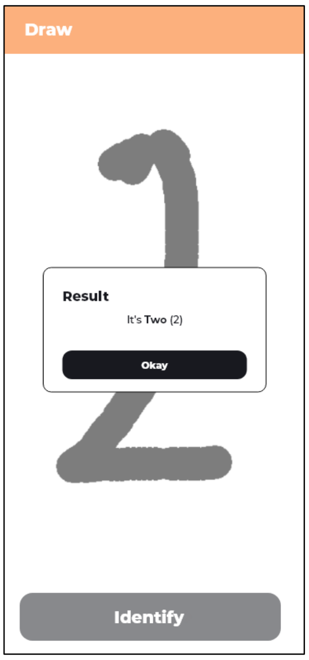


Figure 14. The result is displayed which is received from the API as response.

VIII. CONCLUSION

In this project, the Handwritten Digit Recognition using Deep learning methods has been implemented. The most widely used Machine learning algorithms CNN has been trained and tested on the MNIST dataset. With extensive testing using the MNIST data, the current function suggests the role of various hyper parameters. We also confirmed that a good adjustment of hyper parameters is important in improving the performance of Convolutional Neural Network. Utilizing this deep learning technique, a high amount of accuracy can be obtained. This model is able to achieve a recognition rate of 98.85% accuracy and is significantly identifying real world images as well. The effect of increasing the number of convolutional layers on CNN structure in the performance of handwritten digital recognition is clearly demonstrated by experiments. Later it can be expanded to identify the character and a real-time person's handwriting. Digital recognition is the first step in the larger field of Artificial Intelligence and Computer Vision. As can be seen from the test results, CNN proves to be much better than other classifiers. The results can be made more accurate with multiple layers of convolution and an additional number of hidden neurons. It can completely eliminate the need for typing.

Digital recognition is an excellent model for learning about neural networks and provides a great way to develop the most advanced methods of deep learning. By creating the API, we have made this model open to all. And by the app anyone can use our model without knowing the background processes. The handwritten digit recognition using convolutional neural networks has proved to be of a fairly good efficiency and application for the purpose of introducing and demonstrating neural networks to the general public. The project was a great experience of working in a team. The importance of time bound and coordinated execution of work was realized.

IX. REFERENCES

[1]. Simard, P.Y.; Steinkraus, D.; Platt, J.C. Best practice for convolutional neural networks applied to visual document analysis. In Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), Edinburgh, UK, 3–6 August 2003

[2]. K. G. Pasi and S. R. Naik, "Effect of parameter variations on accuracy of Convolutional Neural Network," in 2016 International Conference on Computing, Analytics and Security Trends (CAST), 2016, pp. 398-403: IEEE

[3]. M. M. A. Ghosh and A. Y. Maghari, "A Comparative Study on Handwriting Digit Recognition Using Neural Networks," 2017 International Conference on Promising Electronic Technologies (ICPET), Deir El-Balah, 2017, pp. 77-81.

[4]. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 39, 2481–2495. CrossRef]

[5]. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.

[6]. Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. 2018, 40, 834–848.

[7]. Y. LeCun, "The MNIST database of handwritten digits," http://yann. lecun. com/exdb/mnist/, 1998.

[8]. Nimisha Jain, Kumar Rahul, Ipshita Khamaru. AnishKumar Jha, Anupam Ghosh (2017). “HandWritten Digit Recognition using Convolutional Neural Network (CNN)”, International Journal of Innovations & Advancement in Computer Science, IJIACS, ISSN 2347 – 8616, Volume 6, Issue 5.

[9]. Berend-Jan van der Zwaag, “Handwritten Digit Recognition: A Neural Network”, International Conference, 7th Fuzzy Days, Dortmund, Germany, October 1-3, 2001.

[10]. S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair, “Handwritten Digit Recognition using Machine Learning Algorithms”, Global Journal of Computer Science and Technology: D Neural & Artificial Intelligence Volume 18 Issue 1 Version 1.0 Year 2018.

[11]. Oleg Kovalyov, “Django REST Framework Tutorial: How to Develop APIs”, Django Stars 23 October 2019

[12]. Carmine Zaccagnino,” Uploading a File to a Server from Flutter Using a Multi-Part (form-data) POST Request “, dev.to DEv Community, Feb 3, 2020.

[13]. Ciresan, D.C.; Meier, U.; Schmidhuber, J. Multi-column deep neural networks for image classification. arXiv 2012, arXiv:1202.2745.

[14]. Niu, X.X.; Suen, C.Y. A novel hybrid CNN–SVM classifier for recognizing handwritten digits. Pattern Recognit. 2012, 45, 1318–1325.

[15]. Ahlawat, S.; Rishi, R. A genetic algorithm-based feature selection for handwritten digit recognition. Recent Pat. Comput. Sci. 2019, 12, 304–316.

Cite this article as :

Sh