International Journal of Advance Research in Science and Engineering
Volume No. 12, Issue No. 07, July 2023
www.ijarse.com
IJARSE
ISSN 2319 - 8354

Application for Handwritten Digits Recognition Based on Convolutional Neural Network

[1]P Subhashini (Assistant Professor), [2]Maragoni Sai Praneeth, [3]Motupalli Tanishq

dept.of Computer Science Engineering
Maturi Venkata Subba Rao Engineering College, Osmania University
Hyderabad, India
subhashini.pallikonda@gmail.com
dept.of Computer Science Engineering
Maturi Venkata Subba Rao Engineering College, Osmania University
Hyderabad, India
maragonisaipraneeth@gmail.com
dept.of Computer Science Engineering
Maturi Venkata Subba Rao Engineering College, Osmania University
Hyderabad, India
motupallitanishq@gmail.com

Abstract—

Handwritten digit recognition is a challenging task that has many applications in data statistics, financial business, mail sorting and other fields. In this paper, we propose a novel system that uses a convolutional neural network (CNN) to recognize handwritten digits written on the gui of the application. To improve work efficiency in paperless offices, a handwritten digits recognition system is necessary. This system plays an important role in large-scale data statistics and financial business. We propose a new handwritten digit recognition system based on convolutional neural network (CNN). The network was trained with standardized pictures to learn the spatial characteristics of handwritten digits. The convolutional neural network updates the network parameters with the data set in MNIST for model training. For model testing, we created a gui which used the trained Model to recognize the numbers written and shows them. Our system demonstrates the advantages of using CNNs for handwritten digit recognition, such as high accuracy, robustness, scalability and simplicity.

Keywords—*machine learning, convolutional neural network, softmax regression algorithm, ReLU.*

I. INTRODUCTION

We are living in a digital era where all the things we do have some digital footprint in it. Digital Learning has helped during tough times like Covid.. Learning is an essential part for a life. It helps a person to grow as an individual.

The Handwritten digit recognition is a traditional research field of pattern recognition that has been studied and developed by experts and scholars since the middle of the last century. One of the problems with high application value is the recognition of handwritten digits, which is discussed in this article. The higher the recognition accuracy, the better, because even slight deviations in the recognition of numbers can cause significant errors that cannot be detected through context. In some cases, this can lead to significant losses, such as when filling out checks and accounts in the financial industry.

Handwritten digit recognition systems play an important role in large-scale data statistics and financial businesses such as industry annual inspection, population census, mail sorting, financial statements, tax statements and checks. The main challenges that affect the accuracy of recognition are various personal writing habits and no logical connection in the digital context. In recent years, deep learning-based handwritten digit recognition systems have been developed that can achieve higher accuracy than traditional methods.

Deep learning-based handwritten digit recognition systems use convolutional neural networks (CNNs) to extract features from images and classify them into different categories. These systems can achieve high accuracy rates because they can learn from large amounts of data and generalize well to new data. However, they require a large amount of labeled data for training and can be computationally expensive.

II. PRINCIPLES OF THE METHOD

A. Survey of Major Area

Numerous researchers blatant their contribution in the field of digit recognition. Based upon their outcome and features weights were assigned and it was executed on character recognition system by Hanmandlu and Murthy [1]. Graves and Schmidhuber[2] used a Hidden Markov Model that used a recurrent neural network. This helped in determining the sequence of characters in scripts that are handwritten. This was implemented for classifying the handwritten Arabic words. This gave an accuracy of about 91%. Multilayer perceptron was used by Pal and Singh[3] for recognizing English characters that are handwritten and an accuracy of 94% was successfully achieved. It was also able to improve the computation time for training the dataset. Neves [4] implemented the Support Vector Machine algorithm and compared this with the MLP algorithm. This recognized the offline handwritten characters with a much better accuracy. The tests and training were done on the standard dataset NIST SD19. Although MLP is a better classifier for all nonlinear classes segmentation, it unfortunately gets trapped in a local minima. This lets the support vector machine get a better accuracy. Younis and Alkhateeb used the MNIST Dataset and extracted features without any pre-processing. It addressed the handwritten OCR problem by implementing a deep neural network. The accuracy achieved was 98.46%. [5] .

Dutt[6] also used the MNIST Dataset. But they also used the Keras and Theano libraries and used multilayer CNN and got an accuracy of 98.7%. Ghosh and Maghari [7] received 98.08% and demonstrated that DNN is the best algorithm, after completing a comparative study of 3 neural networks. But there may be similarity in the shape of digits, and for this reason some amount of error rate will be present. CNN should be the best classifier in this case as compared to the other algorithms such as Support Vector Machine, KNN and the Random Forest Classifier for HDR. [8] implemented a DeepLearning4 j framework which was incorporated with a rectified linear units activation which was never used before. The major aim of the paper was to recognize and determine digits that are handwritten with a higher accuracy and low computation time as well.

CNN is used for fault detection and also for Classification purposes. Recognition of handwritten digits is a big issue of interest in the community of researchers. Many research has already been done on this topic and the flow of papers is still on. This topic is a big interest topic in the field of Machine Learning. Highest accuracy has been achieved using deep learning algorithms and also be using different libraries such as theano and keras. Tensorflow has also been used. This enables the accuracy and performance to be better than other algorithms. Convolutional Neural Network is being used in various areas such as Natural Language Processing, Video analysis and many other research areas as well. Sentiment analysis has also gained traction among researchers and people are trying to do more and more research in this field and move deeper into the research area. The research will go deeper in the coming days as well.

B. Conventional Neural Network (CNN)

Convolutional Neural Network[9] is a family of multi-layer neural networks it is particularly designed for use on two-dimensional data, such as images and videos[10]. Basically, it is influenced by earlier work in time-delay neural networks, which reduce learning computation requirements by sharing weights in a temporal dimension and are intended for speech and time-series processing [11]. It has many hierarchy layers to train in a robust manner. This architecture that leverages spatial and temporal relationships to reduce the number of parameters which must be learned and thus improves upon general feed-forward backpropagation training. It is proposed as a deep learning framework that is motivated by minimal data preprocessing requirements[12].

In CNN, small portions of the image are treated as inputs to the lowest layer of the hierarchical structure. The fully connected layers form a network in this first layer is named as “input layer” and the last layer are named as “output layer” and between these two layers and remaining all are known as “hidden layers”. In the hidden layer, the inputs were passed and the output layer calculates the class probabilities for the classification. A regular neural network performs high computation if the size of data is increased or if the number of layers are increased. in these many parameters were calculated without overfitting results for accuracy[13].

On the other hand, CNN not fully connected in all levels of layers. The neurons in these layer connected to a small region. This encourages the local spatial relationship in the data and the hierarchy features to increase the abstraction from low-level to high-level when multiple were layers are stacked[14]. In these first layers can see only a small portion of the input data and the last layers can see the whole of the input data and draw conclusions from it. In CNN there are three types of layers were used in a convolutional network:

Convolutional layer: In this layer few parameters like a number of filters, size of filters, stride, etc. Small window filter slid along with the dimensions of input data and performs dot products between the values stored in the filter and the input data points[15].

Pooling layer: This layer reduces the dimensionality of the input data which reduces the computations, number of parameters and therefore reduces overfitting. Typically, the pooling layer is inserted between convolutional layers. It discards the activations of previous layers and hence forcing the next convolutional layers to learn from a lmited variety of data [15].

Fully connected layer: In this layer neurons that are connected to all neurons of the previous layer as explained it[15].

C. Activation Function (ReLU)

It is the most widely used activation function. It is the used in almost all the convolutional neural networks and deep learning tasks. The range of this function is from 0 to ∞. The function graph is shown in Figure below.

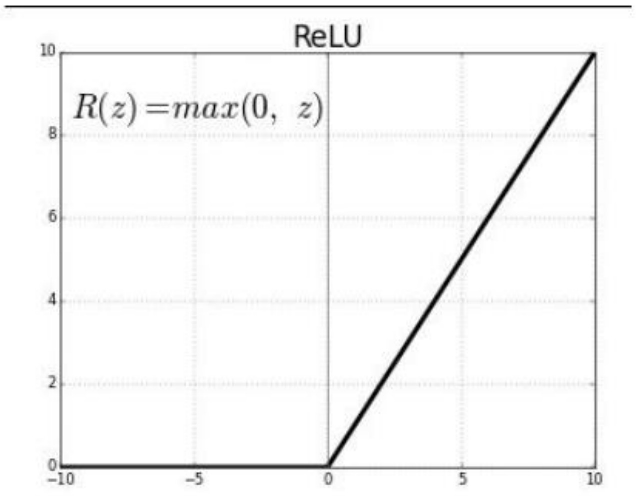


Fig. 1. ReLU Function

D. Softmax function

The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities. If one of the inputs is small or negative, the softmax turns it into a small probability, and if an input is large, then it turns it into a large probability, but it will always remain between 0 and 1.

The softmax function is sometimes called the softargmax function, or multi-class logistic regression. This is because the softmax is a generalization of logistic regression that can be used for multi-class classification, and its formula is very similar to the sigmoid function which is used for logistic regression. The softmax function can be used in a classifier only when the classes are mutually exclusive.

Many multi-layer neural networks end in a penultimate layer which outputs real-valued scores that are not conveniently scaled and which may be difficult to work with. Here the softmax is very useful because it converts the scores to a normalized probability distribution, which can be displayed to a user or used as input to other systems. For this reason it is usual to append a softmax function as the final layer of the neural network.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Fig. 2. Softmax Function

III. DATA SETS

MNIST is a well-known handwritten digit data set, founded by the American Association of Scientists. This data set is often used in the field of machine learning recognition. MNIST contains 10,000 test set data and 60,000 training set data, and each handwritten digital picture is in a 28*28 pixel format. The pictures in this data set are standardized and processed. They are all a 28px*28px grayscale image in the middle. As a common data set, MNIST is often used when testing neural networks. This is also the basic application field of the data set.

Each data unit of the MNIST data set contains two parts (the test data set and the training data set are the same): a picture containing handwritten digits and a corresponding label.

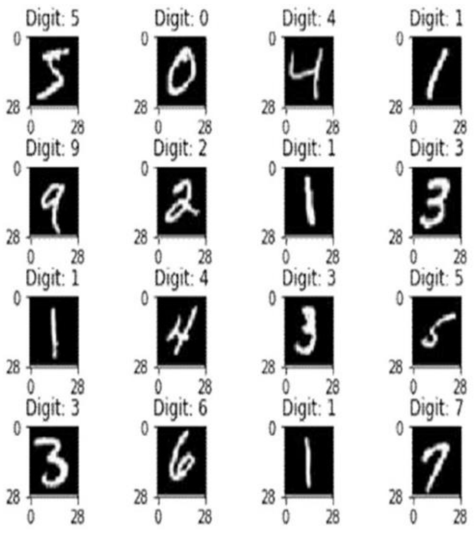
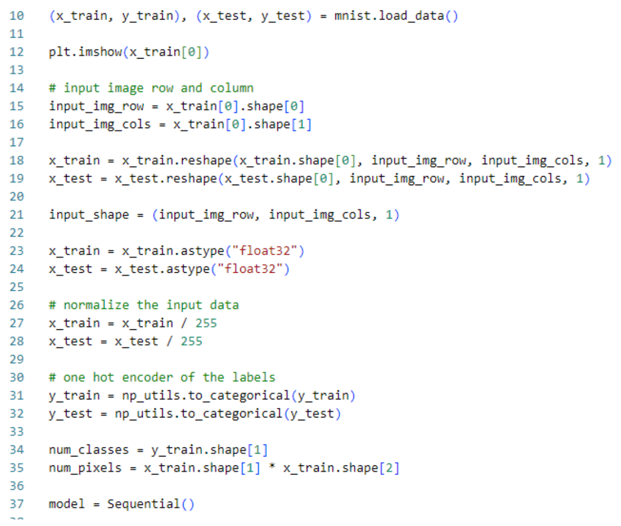


Fig. 3. MNIST Dataset

IV. DIGITAL IDENTIFICATION

A. Pre-processing

When we are required to build a predictive model, we have to look and manipulate the data before we start modelling which includes multiple pre-processing steps such as importing the images, changing the size of the images, changing the colour of the images, visualizing the image dataset and converting them from categorical to vector form.[16] All these steps are combined in total under one thing that is known as Exploratory Data Analysis. We are performing these steps to ease our computing speed and reduce the complexity of the model.



```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

plt.imshow(x_train[0])

# input image row and column
input_img_row = x_train[0].shape[0]
input_img_cols = x_train[0].shape[1]

x_train = x_train.reshape(x_train.shape[0], input_img_row, input_img_cols, 1)
x_test = x_test.reshape(x_test.shape[0], input_img_row, input_img_cols, 1)

input_shape = (input_img_row, input_img_cols, 1)

x_train = x_train.astype("float32")
x_test = x_test.astype("float32")

# normalize the input data
x_train = x_train / 255
x_test = x_test / 255

# one hot encoder of the labels
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)

num_classes = y_train.shape[1]
num_pixels = x_train.shape[1] * x_train.shape[2]

model = Sequential()
```

Fig. 4. Pre-Processing Data

In the reference to Fig 4, we have converted our image from RGB to Gray scale for easy computation by dividing with 255-pixel value to get the value between 0-1.[17] In this process we are also converting the data from categorical value to vector form or binary form.

B. Principle of Convolutional neural network

After we are done with the pre-processing part, the next step is to create the CNN model. CNN stands for Convolutional Neural Network. CNN consists of 4 hidden layers which help in extraction of the features from the images and is able to predict the result.[18] The layers of CNN are (a) Convolutional Layer (b) ReLu Layer (c) Pooling Layer (d) Fully Connected Layer. The Reason we are using CNN is because the fundamental favourable position of CNN contrasted with its archetypes is that it consequently recognizes the significant highlights with no human management.

1. Convolutional Layer

Convolutional layer is a simple application of a filter which acts as an activation function.[19] What this does is takes a feature from a input image, then filter different features from that image and makes a feature map. Some of the features are location, strength etc. the filter is then moved over the whole image and the value of each pixel is calculated.

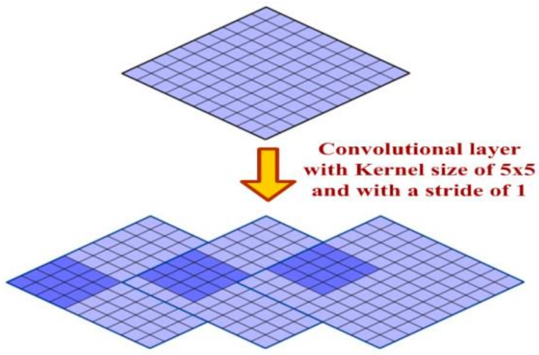


Fig. 5. Representation of kernal and Stride in Convolutional Layer

2. Relu Layer

In simple language, the function of ReLu layer is to remove all the negative pixel values from the image and replace them with zero. This is done to avoid the summing up of pixel values to zero.

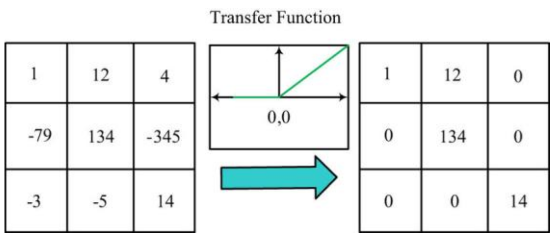


Fig. 6. Relu coverts negative pixel values to zero.

3. Pooling Layer

The main function of this layer is to shrink the image size. This is done to ease the computation speed and also decrease the computational cost.[20] What this layer basically does is takes a matrix of 2 x 2 and stride (movement from one pixel to another) of 1 and move the window to whole of the image. In each of the window the highest value is taken and this process in carried on to each part of the image. Taking an example, if before pooling layer we had a matrix of 4 x 4 then after pooling layer the image matrix will reduce to 2 x 2 matrix.

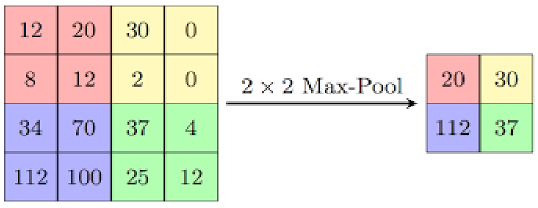
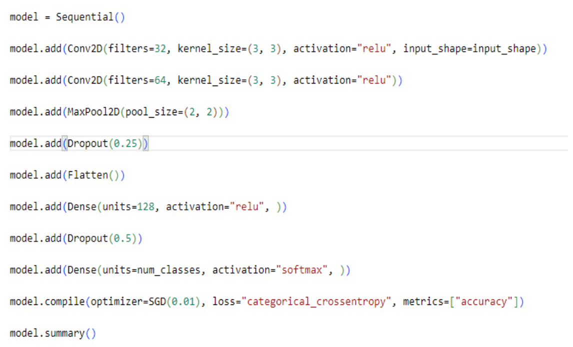


Fig. 7. Representation of 2x2 Pooling

4. Fully connected Layer

This is the last layer of CNN. This is the part where the actual classification happens. All the matrix from the pooling layer is stacked up here and put into a single list. The values which are higher are the points of prediction for the given image.

5. Below is the visualization of the CNN model with all necessary libraries included. Sequential is used to keep all the processes in a sequence one after another. The code Dropout is used to dropout values from the dataset to reduces the over fitting.



```
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3, 3), activation="relu", input_shape=input_shape))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation="relu"))

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

model.add(Dropout(0.25))

model.add(Flatten())

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

model.add(Dropout(0.5))

model.add(Dense(units=num_classes, activation="softmax", ))

model.compile(optimizer=SGD(0.01), loss="categorical_crossentropy", metrics=["accuracy"])

model.summary()
```

Fig. 8. CNN Model

C. Recognition results

After building the model it's time for evaluation of the model. In this part of the project, we must check if the above built model is working to our expectation. But before that we first need to train the model on the evaluation dataset.

V. IMPLENENTATION

In pattern recognition and image processing problems neural network are often used. The most promising tool to carry out this is CNN. It is a deep learning technique which is inspired by the neuron connectivity pattern in animal visual cortex. CNN is mainly used in image recognition, object recognition etc. The CNN model for recognizing the digits is constructed using the KERAS library of python and TensorFlow as a backend. Sequential Model is used as classifier which consists of linear stack of layers.

In CNN the neurons have learnable weights and biases. Compared to others CNN requires minimum preprocessing. The input is always a vector in neural network but in CNN the input will be a multi channelled image. The CNN comprises of an input layer, hidden layers and an output layer. The hidden layer constitutes the onvolutional layer, Rectified layer unit (ReLU) i.e. activation function, pooling layers, normalized layers and fully connected layers.

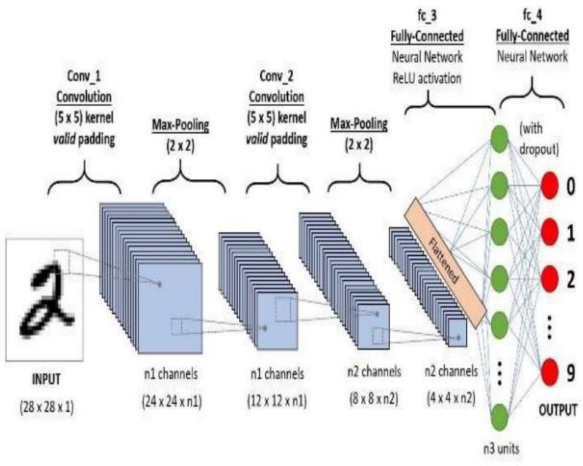


Fig. 9. A seven-layered CNN architecture for digit recognition

The input layer consists of 28 by 28-pixel images which means it contain the network of 784 neurons as input data. The layer next to input layer is convolutional layer which receives the output from the input layer. A convolution operation is performed on the input in convolutional layer. The operation performed in this layer helps in reducing the number of free parameters. And design each filter such that it slides above the input data to have pixels with the utmost intensity.

The Rectified Linear Unit (ReLU) is a activation function which is a linear function, easy to use and achieves better performance. Also, it does not saturate. The concept of pooling layer is to merge the product of neuron at one layer into single neuron at the next level. The fully connected layer reckon the score of input digits. At the end a SoftMax classifier is applied at the end which returns the probabilities of all output classes. From all these values, the class with the largest value is selected as a final classification. To work on Keras API, it is converted into 4 dimensional NumPy arrays. The normalization is done by dividing RGB code by 255. Adam optimizers is used to update the neuron weights and it require little memory.

A. Image Pre-processing

After training the model it will be saved as a classifier. Now the image pre-processing is done on the input image obtained by the interface. The image processing is done with the help of Open Cv2. We are reshaping the data to suit the neural network that is being built .After the reshaping process the data is now converted into categorical data.

The steps taken are:

- Read image: In this, the path of our image is stored into a variable and then a function is created that converts images into array.
- Resize image: Some images captured by camera vary in size. The base size needs to be established for all images fed into our system.

- Remove Noise: By using Gaussian Blur () function image noise may be used.
- Segmentation: More techniques to smoothen and reduce further noise is done.
- Scaling and Padding: The image is then scaled and padded to make it suitable for prediction.

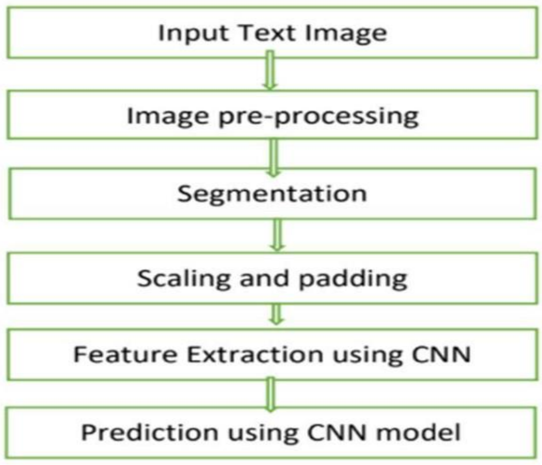
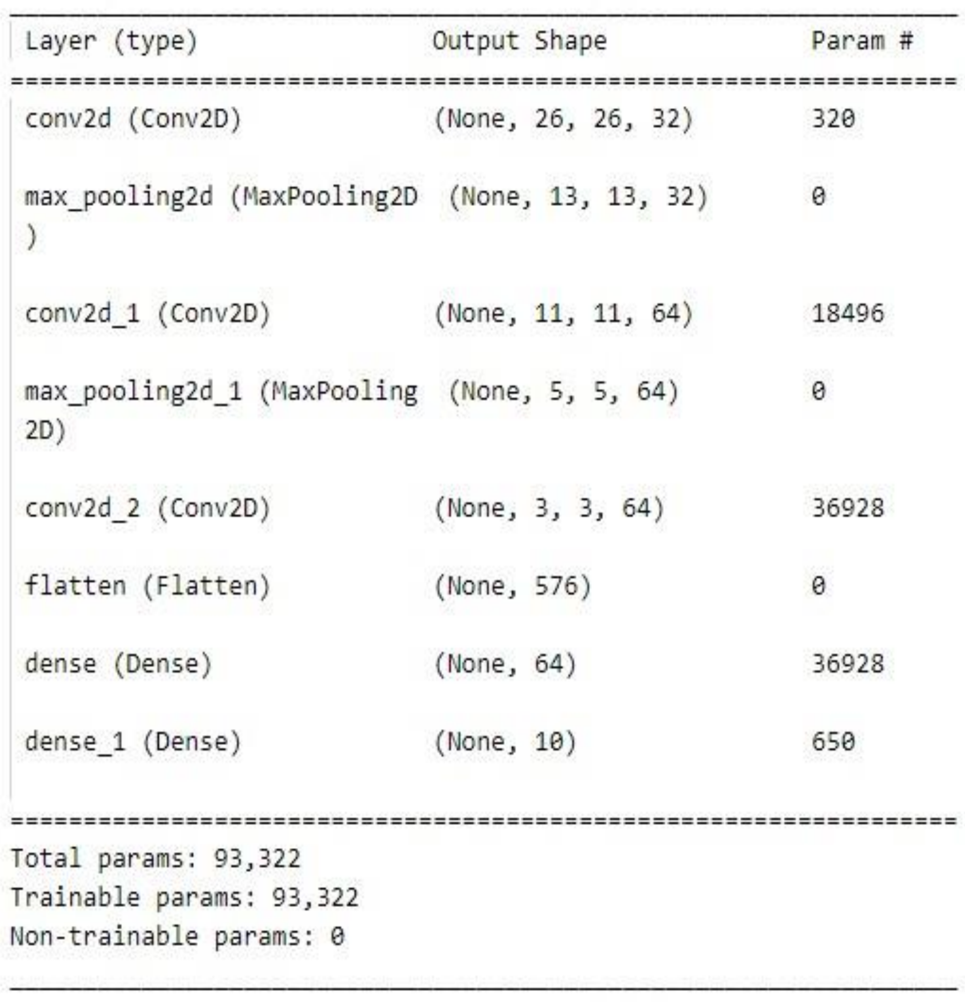


Fig. 10. Flow Diagram of proposed method

VI. EXPERIMENTS

Our experiment is divided into two parts: training and test. For training part, we trained the convolutional neural network. For training part, we use the numbers written by hands and the numbers from the MNIST data set to test the system. For test part, the system obtains handwritten data through the gui created, and constantly refreshes the predicted output in the window. Since the data set we trained is the MNIST data set, all the numbers in it are in the form of white characters on a black background.

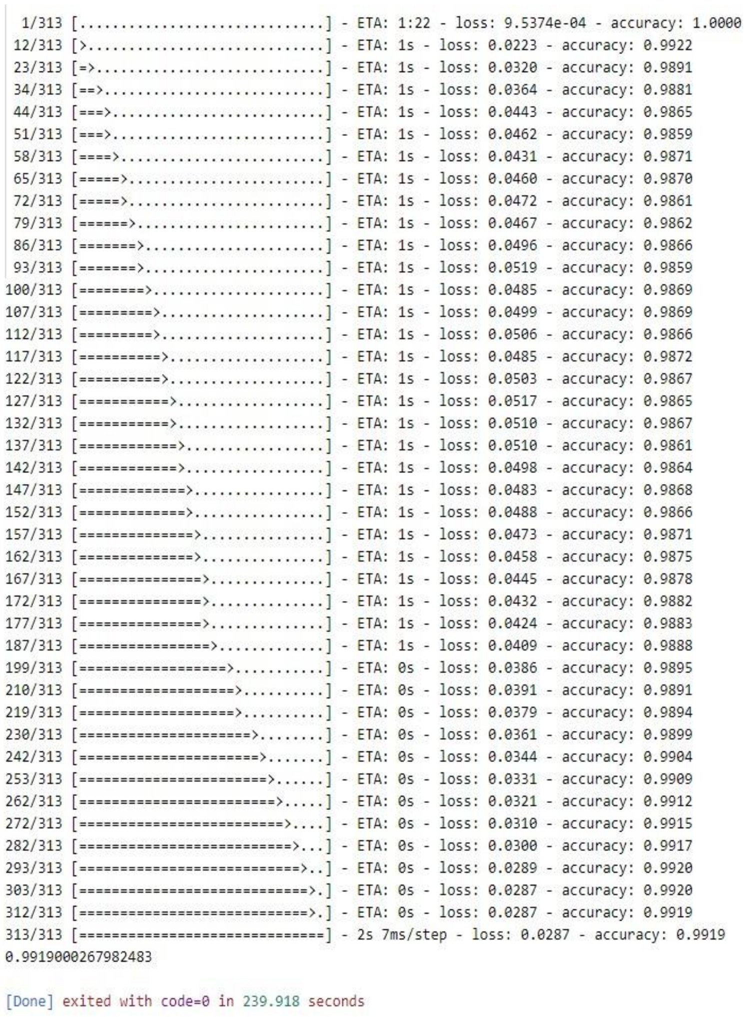
A. Model Creation (CNN Model)



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 64)                36928

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

Fig. 11. It shows the Layers that are used in the Creation of the CNN Model

B. Model Training

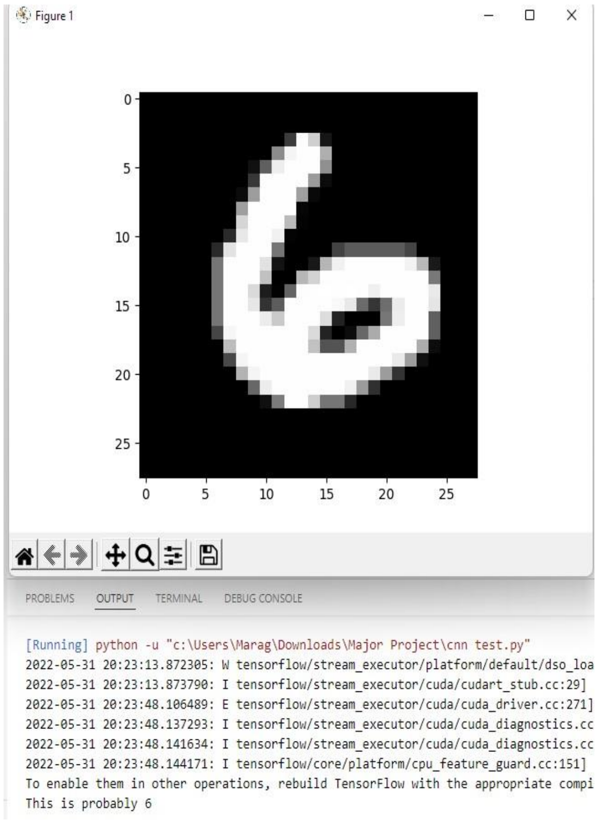


```
  1/313 [..............................] - ETA: 1:22 - loss: 9.5374e-04 - accuracy: 1.0000
 12/313 [>.............................] - ETA: 1s - loss: 0.0223 - accuracy: 0.9922
 23/313 [=>............................] - ETA: 1s - loss: 0.0320 - accuracy: 0.9891
 34/313 [==>...........................] - ETA: 1s - loss: 0.0364 - accuracy: 0.9881
 44/313 [===>..........................] - ETA: 1s - loss: 0.0443 - accuracy: 0.9865
 51/313 [===>..........................] - ETA: 1s - loss: 0.0462 - accuracy: 0.9859
 58/313 [====>.........................] - ETA: 1s - loss: 0.0431 - accuracy: 0.9871
 65/313 [=====>........................] - ETA: 1s - loss: 0.0460 - accuracy: 0.9870
 72/313 [=====>........................] - ETA: 1s - loss: 0.0472 - accuracy: 0.9861
 79/313 [======>.......................] - ETA: 1s - loss: 0.0467 - accuracy: 0.9862
 86/313 [=======>......................] - ETA: 1s - loss: 0.0496 - accuracy: 0.9866
 93/313 [=======>......................] - ETA: 1s - loss: 0.0519 - accuracy: 0.9859
100/313 [========>.....................] - ETA: 1s - loss: 0.0485 - accuracy: 0.9869
107/313 [=========>....................] - ETA: 1s - loss: 0.0499 - accuracy: 0.9869
112/313 [=========>....................] - ETA: 1s - loss: 0.0506 - accuracy: 0.9866
117/313 [==========>...................] - ETA: 1s - loss: 0.0485 - accuracy: 0.9872
122/313 [==========>...................] - ETA: 1s - loss: 0.0503 - accuracy: 0.9867
127/313 [===========>..................] - ETA: 1s - loss: 0.0517 - accuracy: 0.9865
132/313 [===========>..................] - ETA: 1s - loss: 0.0510 - accuracy: 0.9867
137/313 [============>.................] - ETA: 1s - loss: 0.0510 - accuracy: 0.9861
142/313 [============>.................] - ETA: 1s - loss: 0.0498 - accuracy: 0.9864
147/313 [=============>................] - ETA: 1s - loss: 0.0483 - accuracy: 0.9868
152/313 [=============>................] - ETA: 1s - loss: 0.0488 - accuracy: 0.9866
157/313 [==============>...............] - ETA: 1s - loss: 0.0473 - accuracy: 0.9871
162/313 [==============>...............] - ETA: 1s - loss: 0.0458 - accuracy: 0.9875
167/313 [===============>..............] - ETA: 1s - loss: 0.0445 - accuracy: 0.9878
172/313 [===============>..............] - ETA: 1s - loss: 0.0432 - accuracy: 0.9882
177/313 [===============>..............] - ETA: 1s - loss: 0.0424 - accuracy: 0.9883
187/313 [================>.............] - ETA: 1s - loss: 0.0409 - accuracy: 0.9888
199/313 [==================>...........] - ETA: 0s - loss: 0.0386 - accuracy: 0.9895
210/313 [===================>..........] - ETA: 0s - loss: 0.0391 - accuracy: 0.9891
219/313 [===================>..........] - ETA: 0s - loss: 0.0379 - accuracy: 0.9894
230/313 [=====================>........] - ETA: 0s - loss: 0.0361 - accuracy: 0.9899
242/313 [======================>.......] - ETA: 0s - loss: 0.0344 - accuracy: 0.9904
253/313 [=======================>......] - ETA: 0s - loss: 0.0331 - accuracy: 0.9909
262/313 [========================>.....] - ETA: 0s - loss: 0.0321 - accuracy: 0.9912
272/313 [=========================>....] - ETA: 0s - loss: 0.0310 - accuracy: 0.9915
282/313 [==========================>...] - ETA: 0s - loss: 0.0300 - accuracy: 0.9917
293/313 [===========================>..] - ETA: 0s - loss: 0.0289 - accuracy: 0.9920
303/313 [============================>.] - ETA: 0s - loss: 0.0287 - accuracy: 0.9920
312/313 [============================>.] - ETA: 0s - loss: 0.0287 - accuracy: 0.9919
313/313 [==============================] - 2s 7ms/step - loss: 0.0287 - accuracy: 0.9919
0.9919000267982483

[Done] exited with code=0 in 239.918 seconds
```

Fig. 12. The Above Figure Shows the number of times the Model is Trained on Different Parameters with Training data to Increase the Efficiency.

The Model must be trained multiple times on the labelled data so that we can increase the Efficiency of the Model. As the Model is trained multiple times on the data the Efficiency Increases.

C. Model Testing



```
[Running] python -u "c:\Users\Marag\Downloads\Major Project\cnn test.py"
2022-05-31 20:23:13.872305: W tensorflow/stream_executor/platform/default/dso_loa
2022-05-31 20:23:13.873790: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
2022-05-31 20:23:48.106489: E tensorflow/stream_executor/cuda/cuda_driver.cc:271]
2022-05-31 20:23:48.137293: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc
2022-05-31 20:23:48.141634: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc
2022-05-31 20:23:48.144171: I tensorflow/core/platform/cpu_feature_guard.cc:151]
To enable them in other operations, rebuild TensorFlow with the appropriate compi
This is probably 6
```

Fig. 13. Sample Test Case for Testing the Model from Test images in MNIST

The model is tested with Random Test Image present in the Testing Library of the MNIST Dataset. It recognizes the digit in the image and gives it as the output. This is a Sample Test Case 1 used to check if the model is working and predicting the right digit or not.

VII. CONCLUSIONS

In this paper, we have presented a complete system of handwritten digit recognition with high speed and accuracy. Convolutional Neural Network gets trained from the real-time data and makes the model very simple by reducing the number of variables and gives relevant accuracy. In our project, we used CNN with some libraries like Keras, Matplotlib, CV2, Tensorflow to get the maximum accuracy.

It can Recognize multiple digits and show the digit recognized on top of the window created around the digit.

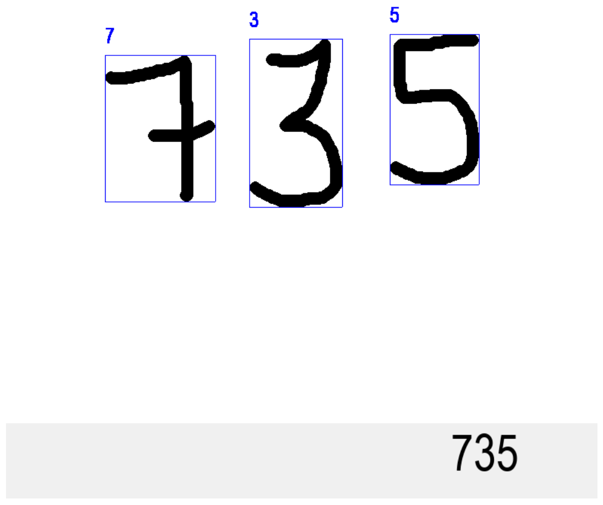


Fig. 14. The Output that is shown in the gui using the trained model

ACKNOWLEDGMENT

The authors are very grateful to anonymous reviewers and suggestions to improve the presentation and theoretical results of this paper.

REFERENCES

[1] Hanmandlu M, Murthy OVR (2007) Fuzzy model-based recognition of hand written numerals. Pattern Recogn 40(6):1840–1854

[2] Graves A, Schmidhuber J (2009) Ofine handwriting recognition with multidi mensional recurrent neural networks. In: Advances in neural information processing systems NIPS’22, vol 22. MIT Press, Vancouver, pp 545–552

[3] Pal A, Singh D (2010) Handwritten english character recognition using neural network. Int J Comput Sci Commun 1(2):141–144

[4] Neves RFP, Filho ANGL, Mello CAB, Zanchettin C (2011) ASVM based of-line handwritten digit recognizer. In: International conference on systems, man and cybernetics, IEEE Xplore, Brazil, 9–12 Oct 2011, pp 510–515

[5] Younis KS, Alkhateeb AA (2017) A new implementation of deep neural networks for optical character recognition and face recognition. In: Proceedings of the new trends in information technology, Jordan, Apr 2017, pp 157–162

[6] Dutt A, Dutt A (2017) Handwritten digit recognition using deep learning. Int J AdvRes Comput Eng Technol 6(7):990–997

[7] Ghosh MMA, Maghari AY (2017) A comparative study on handwriting digit recognition using neural networks. In: International conference on promising electronic technologies, pp 77–81

[8] Ali, S., Shaukat, Z., Azeem, M., Sakhawat, Z., Mahmood, T., & ur Rehman, K. (2019). An efficient and improved scheme for handwritten digit recognition based on convolutional neural network. SN Applied Sciences, 1(9), 1125.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[10] F.-J. Huang and Y. LeCun, "Large-scale learning with SVM and convolutional nets for generic object categorization," in Proc. Computer Vision and Pattern Recognition Conf. (CVPR'06), 2006.

[11] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using timedelay neural networks," IEEE Trans. Acoust., Speech, Signal Processing, vol. 37, pp. 328–339, 1989.

[12] "Convolutional neural network," Wikipedia.

[13] X.-X. Niu and C. Y. Suen, "A novel hybrid CNN–SVM classifier for recognizing handwritten digits," Pattern Recognit., vol. 45, no. 4, pp. 1318–1325, 2012.

[14] M. I. Fanany, "Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM)," in Information and Communication Technology (ICoIC7), 2017 5th International Conference on, 2017, pp. 1–6.

[15] L. Deng and D. Yu, "Deep Learning: Methods and Applications," in Foundations and trends in Signal Processing, 2014, pp 192-387.

[16] Wells, Lee & Chen, Shengfeng&Almamlook, Rabia&Gu, Yuwen. (2018). Offline Handwritten Digits Recognition Using Machine learning.

[17] Burel, G., Pottier, I., &Catros, J. Y. (1992, June). Recognition of handwritten digits by image processing and neural network. In Neural Networks, 1992. IJCNN, International Joint Conference on (Vol. 3, pp. 666-671) IEEE.

[18] 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 2014.

[19] Ahmed, M., Rasool, A. G., Afzal, H., & Siddiqi, I. (2017). Improving handwriting-based gender classification using ensemble classifiers. Expert Systems with Applications, 85, 158-168.

Sadri, J., Suen, C. Y., & Bui, T. D. (2007). A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings. Pattern Recognition, 40(3), 898-919.