DIGIT PREDICTOR FOR HAND WRITTEN DIGITS

Gangavarapu Vivek[1], Gundrapally Tejesh[2] and M Thukaram Reddy[3]
[1,2,3]Department of Information Technology
B V Raju Institute of Technology, Narsapur, Telangana, India

ABSTRACT

Handwritten digits are commonly used in various types of documents, such as checks, forms, and invoices. Recognizing these digits can greatly facilitate document processing and enable automation of tasks such as data entry, archiving, and indexing. Also, Handwritten digits are a fundamental part of early education and are used to teach children basic math concepts. Recognizing these digits accurately can help children learn more effectively and facilitate their transition to more complex math concepts.

This paper proposes a digit predictor for recognizing handwritten digits using a convolutional neural network (CNN). The proposed model is trained on a dataset of labeled handwritten digits and achieves high accuracy in predicting the correct digit. The model is trained using the Modified National Institute of Standards and Technology database (MNIST) dataset. The model is evaluated using various performance metrics and compared with other state-of-the-art models. The results demonstrate the effectiveness of the proposed approach in digit recognition and its potential for use in various applications such as optical character recognition, digital document processing, and handwriting analysis.

Keywords: CNN, MNIST dataset, digit predictor.

I. INTRODUCTION

One of the most significant aspects of modern living is machine learning [1]. It is a technique for analyzing algorithms and creating learning systems that autonomously evaluate large amounts of data to uncover hidden patterns that can be used to make predictions or categorize objects according to their properties. Using pattern recognition and pattern learning, algorithmic models are taught to carry out particular jobs in this branch of AI.

In order to handle data with a grid pattern, such as images, CNN is a form of deep learning model. [2] CNN was created with the animal visual brain in mind and is intended to autonomously and adaptively learn spatial hierarchies of features, from low-level to high-level patterns. Convolution, pooling, and fully connected layers make up the three kinds of layers that make up a standard CNN. The first two layers—convolution and pooling—perform feature extraction, while the third layer—a completely connected layer—maps the extracted features into the end result, such as categorization.

A high-level, open-source Python neural network library is called Keras. It is made to make the process of creating, honing, and deploying deep learning models simpler. With the help of a simple API offered by Keras, users may quickly and efficiently create, configure, and train deep learning models. The layers needed to construct the CNN are provided by Keras.

Integrating Keras, NumPy, and Tkinter is a popular approach for building a digit recognition system. NumPy is used for numerical processing, while Keras is used for building and training the deep learning model, and Tkinter is used for creating a user-friendly graphical user interface (GUI) that allows users to interact with the system.

Digit recognition is a challenging problem in the field of computer vision, yet it is also a fundamental task with many practical applications. Accurately recognizing handwritten digits is important for various applications, such as automatic document processing, character recognition, and signature verification.

II. METHODOLOGY AND IMPLEMENTATION

A. Dataset

Here we are using MNIST dataset which consists of handwritten digits of 70,000 labelled samples. In which 60,000 are used for training and 10,000 are used for testing. Each image is represented in a 28 x 28 pixel bounding box and anti-aliased. The distribution of the digits is as shown in Fig 1.

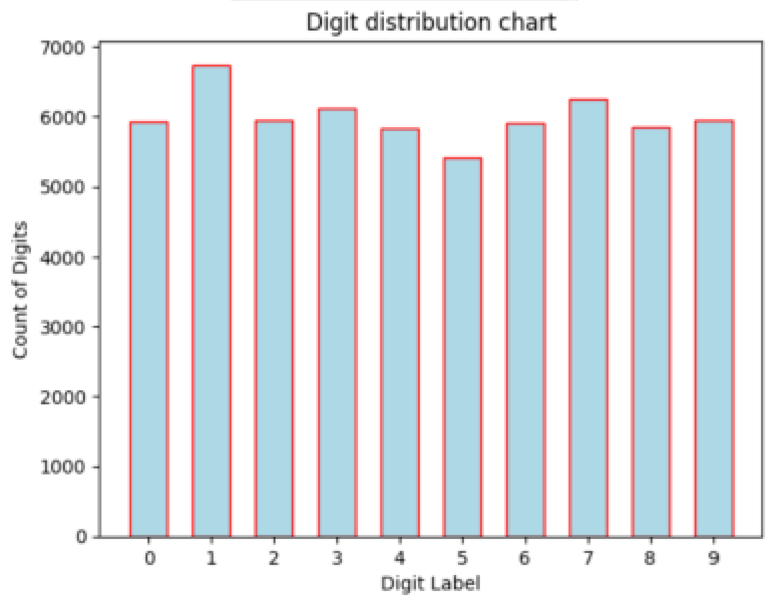


Fig 1. Digit Distribution in MNIST dataset

B. Architecture

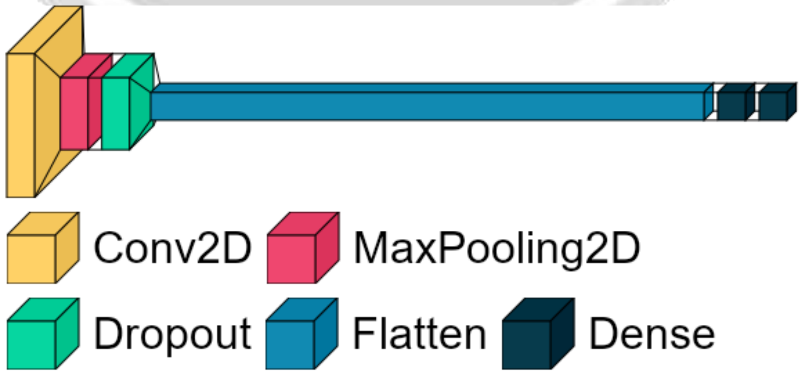


Fig 2. Architecture of CNN [3]

The Fig 2. Shows the architecture of the CNN. Here we are using a Sequential model having the layers that are shown in the above diagram. The first layer is a Convolution2D layer with 28 filters and with the input shape of 28 x 28 x 1. The 28 x 28 is obtained from MNIST dataset and we make it a gray scale image which results in the shape of 28 x 28 x 1. The output of the Convolution2D layer is given to the MaxPooling2D layer having pool size of (2,2) which reduces the output of Convolution2D layer. Then quarter part of nodes are dropped due to the Dropout layer. Later this output is flattened using the Flatten layer, i.e., the 2D layer is converted to 1D layer. Then the output of flatten layer is passed to Dense layer having 128 nodes. At last a Dense layer is added with 10 nodes with SOFTMAX activation function which gives the probability of each node being the classified result. Then the digit having the highest probability will be the predicted digit of the given input. Except for the last layer, the three layers, i.e., Convolution2D layer and the two dense layers are using the RELU activation as it efficiently models the non-linear data. The input for the model is taken from tkinter gui, where the user draws the digit using the mouse pointer. Then the image is pre-processed and will be predicted with the model.

C. Modules

The modules in this system are:

1. Importing required modules and libraries
2. Dataset importing and preprocessing
3. Building model
4. Evaluating model

The explanation of the modules is as follows:

1. Importing required modules and libraries:

Before we start we will import the required modules and libraries. Here we need NumPy module for pre-processing the data, keras module for building the model, matplotlib to visualize and record the observations and visualkeras [3] to visualize the model architecture.

2. Dataset importing and preprocessing:

The MNIST dataset is available in keras module. We will import it as follows:

```
from keras.datasets import mnist
```

Also we can directly load the train and test data separately from functions available in keras module.
This dataset consists image of the shape 28 x 28. We will convert this shape into 28 x 28 x 1, i.e., we will convert the images into gray scale as it removes difficult computational requirements.

3. Building model:

Now we will build the model based on our requirements. We can customize our model based on our dataset, complexity of the problem and computational resources available.The architecture of the model is explained in the previous sections.

4. Evaluating Model:

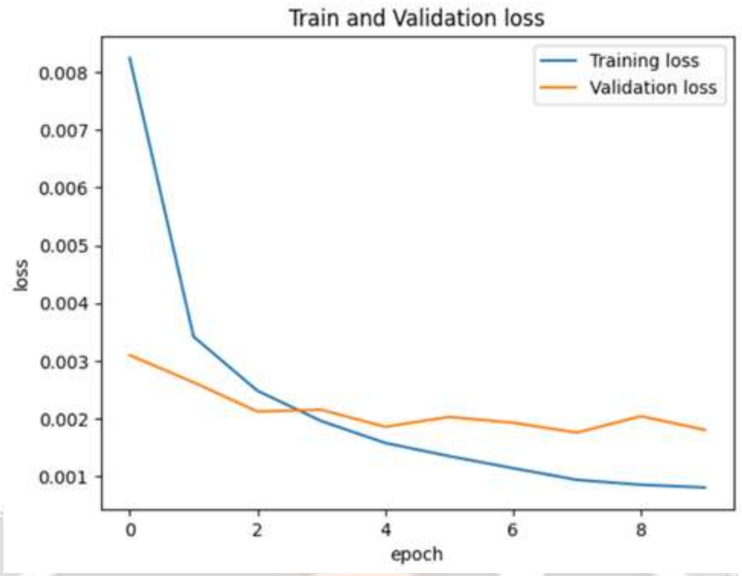


Fig 3. Training and Validation loss

From Fig 3. We can see that the Validation and Training loss are nearly equal which indicates that the model is persistent.

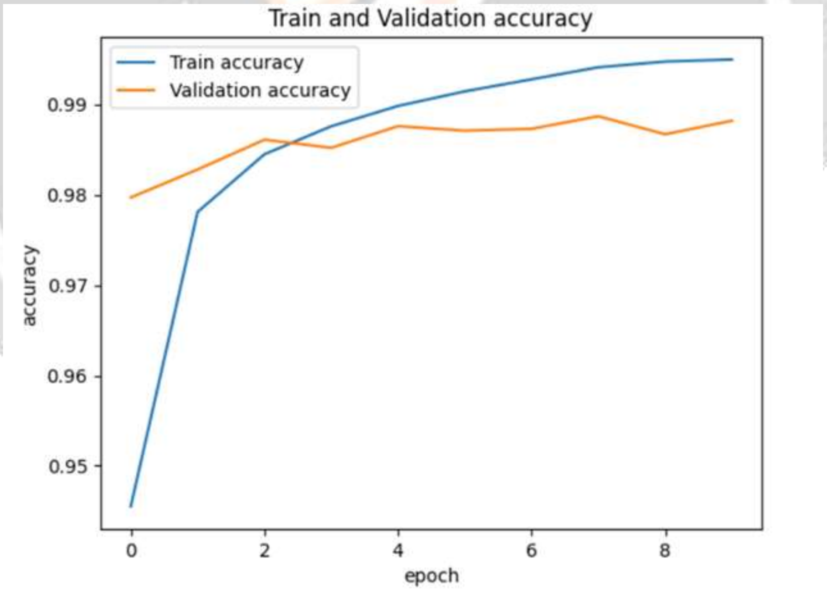
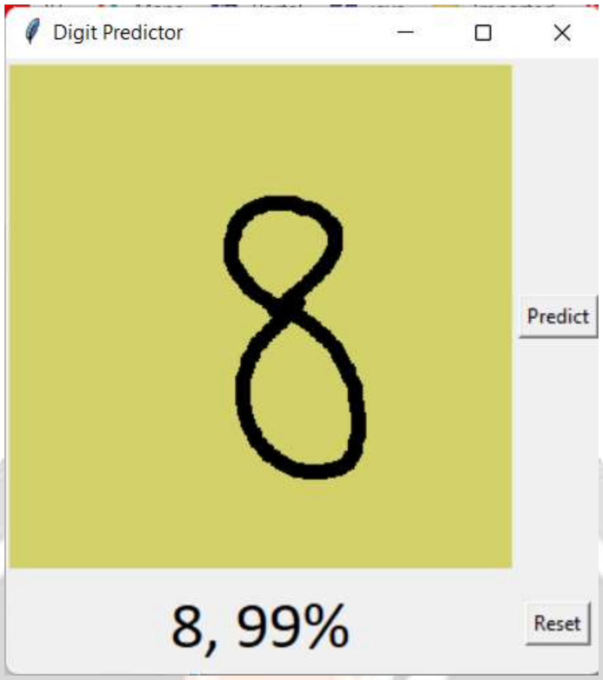


Fig 4. Training and Validation accuracy

From Fig 4. We can see that the Validation and Training accuracy are nearly equal which indicates that the model is persistent.

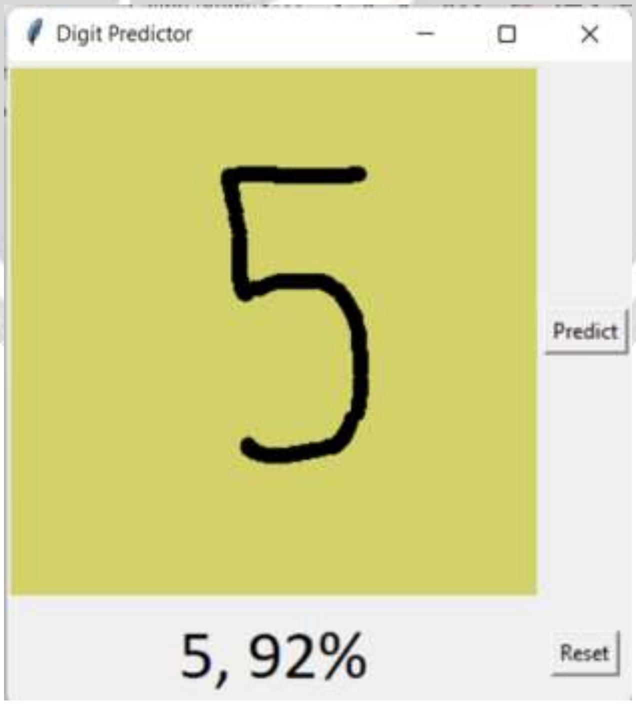
III. RESULTS AND CONCLUSION

A. Results



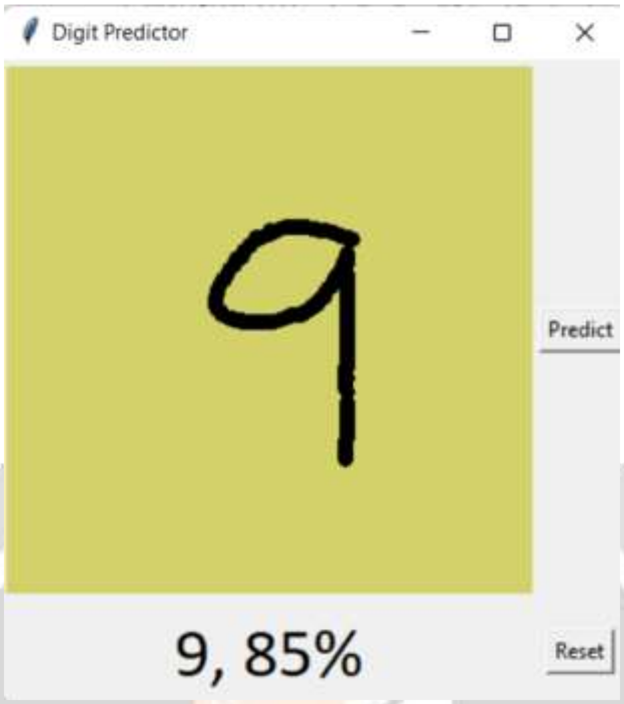
8, 99%

Fig 5. Output for Digit 8



5, 92%

Fig 6. Output for Digit 5



9, 85%

Fig 7. Output for Digit 9

The above images Fig 5., Fig 6., Fig 7. Shows various digit predictions along with their accuracy of the prediction by the previous built model.

B. Conclusion

CNNs are well-suited for image classification tasks, and in this case, the model is trained on a dataset of images of handwritten digits to learn the patterns and features that distinguish different digits. The model is built using the Keras library and Python programming language. The performance of the model is evaluated using metrics such as accuracy and loss, and the epoch vs. loss curve is analyzed to determine the optimal number of training epochs. The above CNN algorithm has the accuracy of 98.8 %.

IV. REFERENCES

[1] Kumar, R. (2022, June 30). Machine Learning. International Journal for Research in Applied Science and Engineering Technology, 10(6), 2460–2460. https://doi.org/10.22214/ijraset.2022.44376

[2] Yamashita, R., Nishio, M., Do, R.K.G. et al. Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9

[3] @misc{Gavrikov2020VisualKeras, author = {Gavrikov, Paul}, title = {visualkeras}, year = {2020},

publisher = {GitHub}, journal = {GitHub repository},

howpublished = {\url{https://github.com/paulgavrikov/visualkeras}},

}