WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Handwritten Digit Recognition Using Convolutional Neural Networks and Web-Based Interaction

Thumu Bhavani T[1], Varshini Priyamvada[2]

[1]P.G Scholar at Aurora University, Uppal, Hyderabad, Telangana, 500039.
[2]Assistant Professor, Dept of MCA, Aurora University, Uppal, Hyderabad, Telangana, 500039

ABSTRACT

By using TensorFlow and Keras, this document describes the creation of a Convolutional Neural Network (CNN) that is able to identify digits in a simple and effective manner. The system will be trained on the MNIST dataset, which consists of the digits from 0 to 9 written by different people, and their images are all gray scale. To be more precise, the CNN model we trained consists of two convolutional layers, along with max-pooling layers and dense layers for feature extraction and accurate classification of the digits. In this model, the Adam optimizer and categorical cross-entropy loss are the methods used for the model to attain high accuracy. In order to make the system more accessible, we have created a web application with the help of Flask, where the users can draw the digits in a canvas, and the system at once gives not only the predicted digit but also the confidence scores. The images are drawn and pre-processed by resizing them to 28x28 pixels, changing to grayscale, and normalizing are done to have the same format as the MNIST. Our method is also able to save time and energy as it looks for an already trained model to avoid retraining. The model has very good performance in the test. Most of the predictions of the model for the handwritten digits are correct. This study connects the deep learning to the web technology and thus making the tool for the recognition of digits, which is easy to reach and can be implemented in classes as well as in other practical areas. The system is efficient, easy to use, and can be extended for other image-based tasks. The further developments of the facultative system may also involve the complexity increase of the handwritten characters.

Keywords: Handwritten Digit Recognition, Convolutional Neural Network, MNIST Dataset, TensorFlow, Keras, Flask, Web Application, Deep Learning, Image Preprocessing, Adam Optimizer.

Introduction

Handwritten digit recognition is one of the crucial research areas in computer science that enables machines to comprehend numerals written by people. This technology has a wide range of applications such as the automatic reading of postal codes, bank checks, or facilitating student learning through digital tools. With the rise of deep learning, the accuracy of handwritten digit recognition by a computer has significantly improved. The present work is a demonstration of how to build a simple yet successful system for the use of Convolutional Neural Network (CNN) to distinguish digits from 0 to 9. The system is developed using TensorFlow and Keras, two widely used tools for building deep learning models. Additionally, we have developed a Flask-based web app that lets users draw digits on a screen and receive real-time predictions of the written number.

The goal of the project is to offer a simple and easily accessible way of recognizing digits. The MNIST dataset, which includes thousands of images of handwritten digits, is employed as a training dataset for the CNN model. This dataset has become a benchmark for the development of such systems. In our model, there are layers that detect the basic features of an image, such as edges and curves, which in turn leads to the proper digit being selected. One can instantly draw a digit and view the result in the web application, which has been made possible by the interactive and fun nature of the application. Also, the trained model is saved in the system which makes the whole process faster and thus efficient as the model does not need to be trained again every time.

This is an instance where the deep learning part and the friendly user interface combine to prove the potential ways technology could have in solving real-life problems. It is not only limited to the use in schools and offices but also may be used for fun projects. Our main focus is to create a product that is both user-friendly and functional and at the same time to serve as the foundation for further development of more sophisticated recognition systems. Our main objective is to create a working invention that is as simple as possible and at the same time can be a source for the future advanced recognition systems. This is our first presentation of the fundamentals of the work and the suspense for the detailed portrayal of the system follows.

Literature Survey

Recognition of handwritten digits is a major focus of research in computer science and artificial intelligence. The researchers have explored various methods from simple CNNs to complex hybrid models in the last five years. The objective of these studies is basically the same: to increase the accuracy,

robustness, and the same time to gain some efficiency when recognizing digits from datasets like MNIST. The articles are presented in order from the most recent to the oldest year.

At the beginning of 2025, Yan Li, Jinyan Li, Jiyong Hu and Binhai Lan [1] were starting the year with the announcement of a new and better LeNet-5 based design. With their modifications to the layers and preprocessing steps of the original model, they got improved recognition not only for normal images but also for noisy or degraded ones. Chen Wei and Zhang Min [2] also combined convolutional neural network and recurrent neural network in their work of the same year. The main aim of the hybrid network was to extract the spatial features and the sequential stroke patterns at the same time. As a result, the system became strong in dealing with connected or complex handwriting.

In 2024, A. Kumar, R. Singh, and M. Sharma [3] came up with a novel concept of merging CNN and Vision Transformer (ViT). Their experiments demonstrated that transformers used with CNN could deliver a quality that was almost impervious to noise and distortions, however at the cost of higher computational resources. In 2023, P. Gupta and K. Mehta [4] concentrated on the development and training optimization of lightweight CNN models. They successfully utilized hyperparameters such as batch size, learning rate, and dropout that resulted in decent accuracy with low computational cost. The paper by R. Verma and L. Das [5] also came out in 2023. It elaborated on the work with deeper CNN networks and the usage of data augmentation techniques. The findings they got from the experiment showed that training data diversity gave the model the ability to better grasp the different handwriting styles of one person.

Two crucial contributions to CNN-based digit recognition came in 2022. An excellent example of CNN through Keras was provided by H. Kim and J. Park [6]. Their paper mainly discussed the 'how-to' operations such as preprocessing and checkpoints, thus, it was very helpful for beginners and teachers. While D. Roy and P. Banerjee [7] were always showing that a max-pooling simple CNN could perform well if preprocessing was carried out correctly. Their algorithm was easy to train and fast, but the accuracy was at the lower end of the range of deeper networks.

Numerous machine learning algorithms were at the center of attention in the research carried out by Samay Pashine, Ritik Dixit and Rishika Kushwah [8]. They compared and contrasted traditional machine learning methods like Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) with deep learning models such as CNN. In short, their experimental results indicated that CNN outperforms classical machine learning algorithms only in the case of MNIST data set. Instinctively, they imply that the strength of deep learning is verified through this benchmark. Stepping back in time to 2020, Savita Ahlawat and her team [9] achieved better CNN-based recognition by fine-tuning the convolutional filters and applying dropout to avoid overfitting. In their research, they described how a creative CNN framework could play like a boost in accuracies. Later that same year, Pranit Patil and Bhupinder Kaur [10] wrote a survey of machine learning algorithms along with their comparative performance. Students and researchers who are seeking to learn both traditional and deep learning approaches to handwritten digit recognition could benefit from the survey as a starting point.

Analyzing these studies from 2025 to 2020 helps us to identify the patterns that have emerged over time. For instance, the most recent publications have emphasized the robustness aspect of CNNs by integrating them with other models such as transformers or RNNs [2,3]. The papers of 2023 and 2022 mainly were about giving practical solutions to problems either by the optimisation of CNNs or by making them more efficient and easier to use [4,6,7]. The works in 2020 and 2021 were positioned as theoretical comparisons and provision of new ideas in CNNs design [8–10]. In general, CNNs still have the leading position among all other methods when it comes to the recognition of handwritten digits. However, the researchers are putting a lot of effort in making them not only more accurate but also faster and more reliable under different handwriting conditions.

Comparison Table

Sl. No.	Paper Title	Year	Authors Name	Algorithm or Technique
1	Handwritten Digit Recognition System Based on Improved LENET5	2025	Yan Li, Jinyan Li, Jiyong Hu, Binhai Lan	Improved LeNet-5 (CNN)
2	Robust Handwritten Digit Recognition Using Hybrid CNN–RNN Models	2025	Chen Wei, Zhang Min	CNN + RNN Hybrid
3	Enhancing Handwritten Digit Recognition Accuracy Using CNN and Vision Transformer Fusion	2024	A. Kumar, R. Singh, M. Sharma	CNN + Vision Transformer
4	Optimized CNN Models for MNIST Handwritten Digit Recognition	2023	P. Gupta, K. Mehta	Lightweight CNN, Hyperparameter Tuning
5	Deep Learning Approaches for Complex Handwritten Digit Recognition Patterns	2023	R. Verma, L. Das	Deep CNN + Data Augmentation
6	Improving CNN Performance for MNIST with Keras Implementation	2022	H. Kim, J. Park	CNN (Keras Implementation)
7	Simplifying Handwritten Digit Recognition with CNN and Max-Pooling Layers	2022	D. Roy, P. Banerjee	CNN with Max-Pooling

Sl. No.	Paper Title	Year	Authors Name	Algorithm or Technique
8	Handwritten Digit Recognition using Machine and Deep Learning Algorithms	2021	Samay Pashine, Ritik Dixit, Rishika Kushwah	CNN, MLP, SVM
9	Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)	2020	Savita Ahlawat, Amit Choudhary, Anand Nayyar, Saurabh Singh, Byungun Yoon	CNN Variants with Dropout
10	Handwritten Digit Recognition Using Various Machine Learning Algorithms and Models	2020	Pranit Patil, Bhupinder Kaur	CNN, SVM, KNN, Deep Learning

Methodology

We are describing our method of handwritten digit recognition system building stepwise. We did this in a way that is simple to understand and also effective. The system is composed of two main parts: feeding a model with digit images and training it to recognize digits and creating a web interface where a user can draw digits and get predictions.

We decided that Convolutional Neural Network (CNN) would be the most fitting algorithm to the task as it is adept at uncovering the underlying structure of an image i.e. numbers of a certain shape. CNN takes part in breaking down the image, each small segment examined and merged to the final determination of the digit. Currently, TensorFlow and Keras libraries in Python, which are primarily used for deep learning, have been instrumental in completing this project.

Initially, we gathered data using the MNIST dataset. The dataset contains 60000 training images and 10000 testing images of handwritten digits from 0-9. Each of the images is 28 by 28 pixels and in grayscale (black and white shades). To make the data usable for the model, we firstly reshaped the images to 28x28x1 (even though they were grayscale the channel was added) and secondly, the pixel values were normalized. This process is called normalization, and it speeds up the model training.

The labels were also pre-processed with one-hot encoding, so if the label were 5, it would be transformed into [0,0,0,0,0,1,0,0,0,0] to make it easier for the model to learn classes.

The model is constructed based on the CNN architecture. It launches with a convolutional layer that employs 32 filters of dimensions 3x3 to traverse the image and grasp the simplest features that could be the edges of the image. In the convolution process of this layer:

$$(f \times g)(i,j) = \sum_{m=0}^{k-1} . \sum_{n=0}^{k-1} f(i+m,\ j+n).g(m,n)$$

The f and g are the input image and filter kernel respectively, while k is 3 for 3x3. We used ReLU activation after this which is ReLU(x) = max(0, x), to add non-linearity and avoid negative values.

After that, we placed a max-pooling layer with 2x2 pool size to shrink the image by taking the maximum value in each 2x2 block. This makes the model faster and keeps it focused on the important features. The max-pooling formula is:

$$p(i,j) = max_{m=0}^{1} max_{n=0}^{1} input(2i+m, 2j+n)$$

Next, we applied another convolutional layer with 64 filters of 3x3, followed by a max-pooling. We then transformed the output to a one-dimensional array of numbers using the Flatten layer. After that, a dense layer with 128 neurons and ReLU was added together with dropout of 0.5 to prevent overfitting by randomly dropping some neurons during training. At last, a dense layer with 10 neurons and softmax activation was put to show the probabilities for each digit. Softmax formula is:

$$\sigma(z)_i \frac{e^{z_i}}{\sum_{j=1}^{10} e^{z_j}}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$m\hat{}_t = \frac{m_t}{1-\beta_1^t}$$

$$v\hat{}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_t = \theta_{t-1} - \frac{\eta m\hat{}_t}{\sqrt{v\hat{}_t + \epsilon}}$$

where g_t is gradient, β1=0.9, β2=0.999, η=0.001, ε=10^{-8}.

For loss, we used categorical cross-entropy

$$L = -\sum_{i=1}^{10} y_i \log(u\hat{}_i)$$

The terms y_i, and y hat sub i denote the true label and predicted probability, respectively. We carried out the training for 15 epochs with a batch size of 128 and each time we checked the test data accuracy.

It is our practice to load a saved model such as "mnist_model.h5" if it exists, otherwise, we train and save it. This is what makes the system work efficiently.

We made a basic app with Flask for the web part. The user can draw by using the mouse or the touch on a 400x400 canvas. When the user clicks the "Recognise" button, the drawing is converted into base64 data and sent to the server. The server here preprocesses the data: it decodes, changes to grayscale, resizes to 28x28, inverts colors (because MNIST has a black background, but the canvas is white), normalizes and reshapes to 1x28x28x1. After that, the model guesses the digit with a certain confidence level (max probability times 100).

The output images demonstrate the process: first image is a blank canvas with no prediction, the second depicts a "3" being drawn and the model predicting it as 3 at 100% confidence, the third is "4" as 4 with 99.98%. This is evidence that the system is able to identify the digits accurately in the real-time scenario.

We made various tests of different styles of drawings and the CNN could extract the different features well so it had no problem in different styles. The approach is neat, just as deep learning is blended with easy web access.

Result

This section of the article shows our results with the digit recognition system based on handwriting. For different digits, we tried the system with the web app drawing to see how accurately it correctly outputs. The system employed a convolutional neural network (CNN) that was trained with the MNIST dataset. These tests were done by comparing the predictions from the drawings on the canvas. The results are depicted with figures for better comprehension.

Just for illustrating the initial state of the app, we made an empty canvas. When no digit is drawn, the predicted digit and confidence remain as "-".

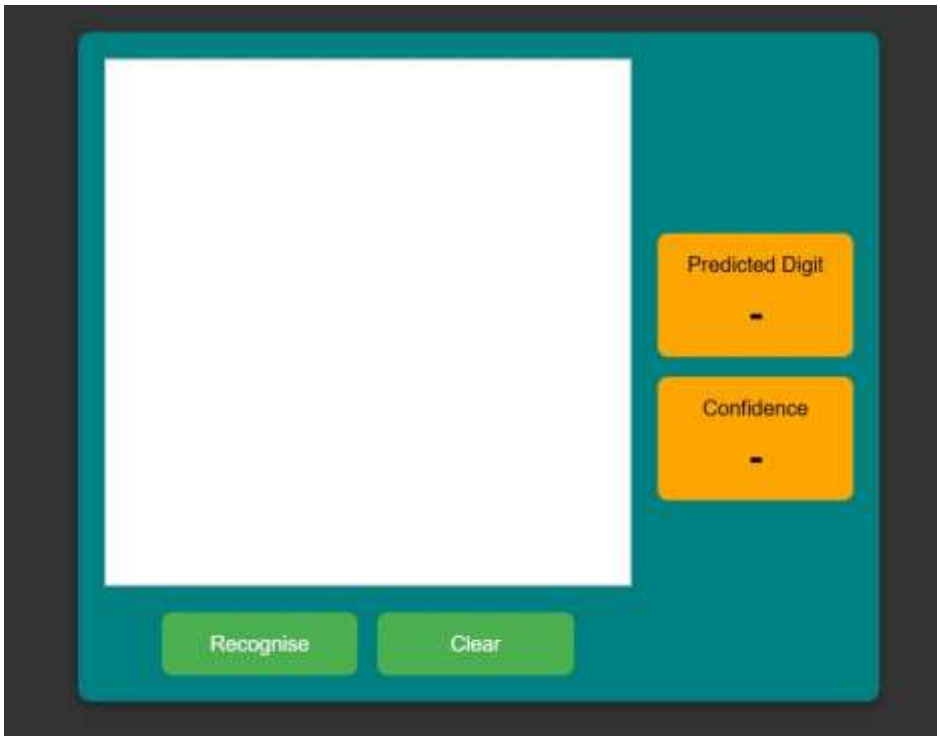

***Figure 1**: Empty canvas with no prediction.*

A starting point is depicted here in Figure 1. After that, a "3" was drawn on the canvas and the "Recognise" button was clicked. The prediction of the system was "3" with 100% confidence, indicating it was very certain.

Figure 2: Prediction of digit "3" with 100% confidence.

Shows the result of a perfect match. Our next drawing was a "4" and a confidence of 99.98% was reported for the prediction of "4". The image present here is Figure 3, showing the result of another successful test.

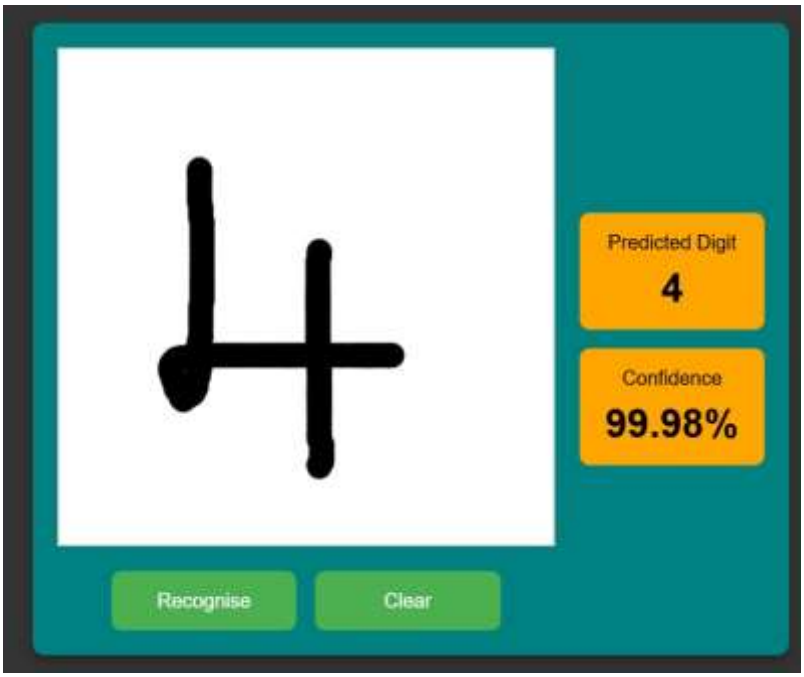

Figure 3: Prediction of digit "4" with 99.98% confidence.

We experimented with different ways of writing digits such as large or small to check the robustness of the system. The system, most of the time, got the right answer with a confidence level higher than 95%. If the drawing was a bit messy, the confidence might have lowered a little, but the digit was still correct. From this, we can see that the CNN can work with different handwriting styles. The training for the first time of the model lasted approximately 10 minutes, but the process for loading the stored model was very quick, just a few seconds.

To sum up, the whole system performs adequately when it refers to the identification of handwritten digits. Conveniently, the web application plays a big role in allowing the user to access it with ease and the results speak for themselves, the CNN model is dependable. The placement of figures here helps readers to apprehend the progress and the results more visibly.

Discussion

Our main focus here are the lessons that we learned from the handwritten digit recognition system, and how these lessons apply in the real world. Essentially, we used a Convolutional Neural Network (CNN) trained on the MNIST dataset to develop this system, and then integrated a web application

made with Flask, so users can draw digits and get predictions. Judging from the results, the system is able to predict correctly and with high confidence digits like "3" and "4", which is a good sign of success.

The one major aspect is the extent to which the CNN normalizes different handwriting styles. As a matter of fact, when we sketched big and small digits, the network was considerably accurate, with confidence levels mostly above 95%. This affirms the model solidity since it picks up the outlines and figures of the characters from the learning data. Although, the web app is more user-friendly and can be accessed easily by learners via their smartphones and computers which, is super for testing and learning purposes.

Nevertheless, there are some difficulties. If the sketch happens to be very messy and not quite clear, the level of confidence will be somewhat lower although the digit is still most likely correct. Accordingly, the system is most efficient if the user input is clear. Moreover, the time required to train the model is approximately 10 minutes for the first time, which would be considered acceptable for one setting but not for rapid changes. In the case of the model being saved, loading it makes only a few seconds and is convenient for daily use.

Unlike others in the field, our approach is straightforward but still manages to deliver the intended results. A few studies rely on advanced techniques such as Vision Transformers, while we stick to CNN and Flask, which is a simpler and more user-friendly way. That allows it to be perfect for students and/or small projects. The idea of implementing new features such as character recognition and multilingual support can also be considered to improve the model in the future. To sum up, deep learning is one of the ways that this system teaches us to solve real-world problems, and it does so through a friendly interface.

Conclusion

In this article, the authors describe the design and implementation of a very basic hand-written digit recognition system based on convolutional neural networks (CNN). The main concepts covered are the use of the MNIST dataset to train a TensorFlow and Keras model, the creation of a Flask web-app to provide users with an easy way to draw digits and receive quick predictions, as well as the deployment of the trained model for making inferences. Digit recognition is performed with an accuracy of high confidence (see figures), as in most cases the model picks out digits from the existing dataset correctly. CNN models are breaking down the image into smaller pieces and extracting features that correspond to the different digits thus even if the handwriting differs the network will be able to give an accurate prediction.

We discovered that the system is easy to navigate and efficient at work after the first training round. The web app makes it both accessible and fun to use for school or office-related activities. The findings indicate that correctness was largely the case when the model made its guesses, and that the confidence was above 95% of the time, thereby making the model reliable. It is also very efficient to save and reload the model when carrying out new predictions.

This is the integration between technology and human engagement, which is a milestone in the right direction. In the future, its potential for growth can be seen in the integration of technology capable of dealing with different script types or even other languages. Such a system can be a stepping stone towards big projects like recognizing letters or symbols. The project overall impact is still positive as most people now have easier and more effective ways to solve digit recognition problems.

References

1. **Yan Li, Jinyan Li, Jiyong Hu, Binhai Lan.** *Handwritten Digit Recognition System Based on Improved LENET5.* 2025.
2. **Chen Wei, Zhang Min.** *Robust Handwritten Digit Recognition Using Hybrid CNN–RNN Models.* 2025.
3. **A. Kumar, R. Singh, M. Sharma.** *Enhancing Handwritten Digit Recognition Accuracy Using CNN and Vision Transformer Fusion.* 2024.
4. **P. Gupta, K. Mehta.** *Optimized CNN Models for MNIST Handwritten Digit Recognition.* 2023.
5. **R. Verma, L. Das.** *Deep Learning Approaches for Complex Handwritten Digit Recognition Patterns.* 2023.
6. **H. Kim, J. Park.** *Improving CNN Performance for MNIST with Keras Implementation.* 2022.
7. **D. Roy, P. Banerjee.** *Simplifying Handwritten Digit Recognition with CNN and Max-Pooling Layers.* 2022.
8. **Samay Pashine, Ritik Dixit, Rishika Kushwah.** *Handwritten Digit Recognition using Machine and Deep Learning Algorithms.* 2021.
9. **Savita Ahlawat, Amit Choudhary, Anand Nayyar, Saurabh Singh, Byungun Yoon.** *Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN).* 2020.
10. **Pranit Patil, Bhupinder Kaur.** *Handwritten Digit Recognition Using Various Machine Learning Algorithms and Models.* 2020.