Proceedings of CONF-CDS 2025 Symposium: Data Visualization Methods for Evaluation
DOI: 10.54254/2755-2721/2025.PO24694

Handwritten Digit Recognition Based on TensorFlow Framework

Yifan Li

North China University of Technology, Beijing, China
3323151601@qq.com

Abstract. This study investigates the widespread application of convolutional neural networks (CNNs) in the field of handwritten digit recognition. To address the challenge of style variability in handwritten digit datasets, an efficient recognition model is proposed. The model is developed using the TensorFlow framework and enhances feature extraction and classification performance through the integration of convolutional layers, pooling layers, ReLU activation functions, batch normalization, and dropout techniques. The MNIST dataset is employed for experimentation. The model is trained using the Adam optimizer and achieves an accuracy of 98.75%, demonstrating the high effectiveness of CNNs in handwritten digit recognition. Furthermore, the study examines the model's limitations in recognizing dynamically sized images and complex mathematical expressions and proposes potential improvements, such as learning rate adjustment and image size standardization.

Keywords: computer vision, convolutional neural network (CNN), handwriting recognition

1. Introduction

With the development of convolutional neural networks (CNNs), handwritten digit recognition has become increasingly important. It is a critical task in the fields of document analysis and pattern recognition, with applications in digital note-taking and educational software. The ability to accurately recognize handwritten digits is essential for extracting human-written content. However, this task poses significant challenges due to the variability in individual handwriting styles. The advent of deep learning, particularly CNNs, has revolutionized image recognition tasks. CNNs automatically learns the spatial hierarchy of the input data through local receptive fields, weight sharing and hierarchical feature extraction. CNNs have demonstrated outstanding performance in various image classification and object detection problems due to their capacity to automatically learn hierarchical features from raw pixel data. This makes them particularly well-suited for handwritten digit recognition. This study employs the TensorFlow framework to build a handwritten digit recognition model designed to classify images from the MNIST dataset. The structure of this paper is as follows: the literature review section summarizes prior research in the field of handwritten digit recognition; the data and methodology section outlines the model architecture, parameters, and experimental results; and the discussion section presents potential improvements to the proposed model.

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.PO24694



Figure 1: Partial sample of MNIST dataset

2. Literature review

In previous studies, many researchers have chosen the TensorFlow framework to build their models. Li and Gao [1] improved the original LeNet-5 model by reducing the number of neurons and parameters, significantly shortening the training time. They conducted experiments on the MNIST handwritten digit dataset using batch input to ensure sufficient network training. Their findings indicated that smaller batch sizes led to higher recognition rates, and when the learning rate was set to 1.0, the mean squared error declined rapidly. Huang et al. [2] constructed both Softmax and CNN models for handwritten digit recognition, with the CNN based on the LeNet-5 architecture. Their experiments on the MNIST dataset revealed that the CNN model achieved a significantly higher recognition accuracy (99.17%) compared to the Softmax function, representing a 7.6% improvement. Tang et al. [3] trained their model using LeNet-5, reducing the output dimension to 10 through Softmax for classification. After 50 training iterations on the MNIST dataset, their model achieved a validation accuracy of approximately 98%. However, misclassifications still occurred, particularly when digits were written with connected strokes or discontinuous pen movements. Ji [4] implemented the LeNet-5 model using a Sequential structure, incorporating two convolutional and pooling layers, followed by two fully connected layers and a final Softmax activation function for classification. This approach achieved a recognition accuracy of 97%, demonstrating strong performance. Some researchers have also combined CNNs with other optimization algorithms. Dutta [5] proposed an enhanced structure that integrated CNN and RNN models, utilizing synthetic data for pre-training to effectively initialize the network. Techniques such as image normalization for skew correction and domain-specific data augmentation were employed to improve invariance learning. This method reduced word and character error rates by approximately 55% and 44%, respectively.

Yang et al. [6] integrated convolutional neural networks (CNNs) with support vector machines (SVMs) and optimized the SVM using a particle swarm optimization (PSO) algorithm. This hybrid approach further enhanced recognition performance. The experiments utilized the Semeion handwritten digit dataset provided by UCI, as well as the MNIST handwritten digit dataset for network training. The convolutional kernel size was set to 5×5, and the pooling window size to 2×2. After determining the optimal PSO parameters c and g, the final recognition rates on the two test sets were approximately 99.11% and 96%, respectively. Deepa [7] employed an eight-layer CNN model with the Adam optimizer, which adaptively adjusts learning rates for each parameter. Compared to traditional stochastic gradient descent, this method achieved faster convergence and better training performance. The model attained 98.33% accuracy on the MNIST validation set and 97.07% on the training set. The study also compared performance between 10 and 20 epochs to

DOI: 10.54254/2755-2721/2025.PO24694

identify the optimal balance between training efficiency and model accuracy. Srivastava [8] trained a model using the Adam optimizer and cross-entropy loss on the MNIST dataset. To enable automatic feature extraction, the neural network was trained using backpropagation, gradient descent, and ReLU activation. Utilizing the DL4J architecture, the model achieved 98% accuracy after 50 epochs. Zhang et al. [9] combined CNN-based feature extraction with incremental learning using Incremental Support Vector Machines to propose a cross-domain, writer-adaptive handwritten digit recognition method. This approach demonstrated significant improvements in small-sample and diverse-style handwriting recognition. Compared to traditional writer-adaptive methods, their solution achieved higher classification accuracy while maintaining strong generalization and adaptability to various writing styles. This advancement contributes to the broader adoption of air handwriting recognition technology in practical applications.

A review of recent research in handwritten digit recognition reveals that convolutional neural networks (CNNs) have achieved significant advancements in this field. Researchers have consistently enhanced recognition accuracy and model efficiency by refining network architectures, integrating optimization algorithms, and incorporating complementary machine learning techniques. With continued progress in deep learning, handwritten digit recognition is expected to be applied in an even broader range of real-world scenarios.

3. Data and methodology

This study employs a Convolutional Neural Network (CNN) for handwritten mathematical symbol recognition. The model architecture consists of essential convolutional and pooling layers, along with several additional functional components, as detailed below:

3.1. Convolutional layer

The system receives the raw image matrix and performs standardized preprocessing, normalizing the data to a range between 0 and 1. In the subsequent convolutional layer, based on the predefined kernel size, stride, and padding, the convolutional kernel slides across the data to compute dot products of local regions, thereby extracting features such as edges and textures. The output of a single convolutional kernel is defined as:

$$O\left(i,j\right)=\sum_{m=0}^{k_h-1}\sum_{n=0}^{k_w-1}\sum_{c=0}^{C_{in}-1}X(i\bullet s_h+m,j\bullet s_w+n,c)\bullet K(m,n,c)+b(1) \tag{1}$$

Where: X is the input tensor, K is the convolution kernel, k is the kernel size, s is the stride length, b is the bias term.

3.2.Activation function

A nonlinear function is introduced to enable the network to approximate complex patterns. Although sigmoid functions were widely used in earlier studies, they are prone to the vanishing gradient problem. The gradient of ReLU in the positive interval is constantly 1, avoiding the problem of gradient attenuation layer by layer in deep networks. In addition, the ReLU function only requires comparison and maximum value operations, and its calculation speed is much faster than that of the activation function involving exponential operations. In this study, the ReLU function is adopted as an alternative. The ReLU function is defined as:

DOI: 10.54254/2755-2721/2025.PO24694

$$f(x) = max(0, x) \tag{2}$$

3.3.Pooling layer

The pooling layer reduces computational load and the number of parameters by reducing the size of the feature map and lowering the model's sensitivity to the training data, the occurrence of overfitting can be prevented. Two common strategies are max pooling and average pooling. Max pooling extracts the maximum value within local regions to retain the most salient features:

$$O(i, j) = maxX(i \bullet s + m, j \bullet s + n) \tag{3}$$

Average pooling, on the other hand, computes the mean value within local regions to smooth features:

$$O\left(i, j\right) = \frac{1}{k_h + k_w} \sum_{m,n} X(i \bullet s + m, j \bullet s + n) \tag{4}$$

This study employs max pooling.

3.4.Fully connected layer

This layer integrates global features and maps them to category labels. It outputs the class with the highest probability to complete the classification. To prevent overfitting, a Dropout layer is added during training, which randomly deactivates neurons with a certain probability.

3.5.Output layer

The Softmax function is used to produce a probability distribution over the output categories. It is defined as:

$$P\left(y\middle|x\right) = \frac{e^{h(x, y_i)}}{\sum_{j=1}^{n} e^{h(x, y_i)}} \tag{5}$$

The convolutional neural network (CNN) model in this study is composed of the aforementioned layers. The specific architecture is as follows:

DOI: 10.54254/2755-2721/2025.PO24694

Table 1: Architecture of CNN

Layer	Specific Architecture
Convolutional Layer	Filters = 32, Kernel Size = 5
Activation Function	ReLU
Pooling Layer	Pool Size = (2, 2)
Convolutional Layer	Filters = 64, Kernel Size = 5
Activation Function	ReLU
Pooling Layer	Pool Size = (2, 2)
Fully Connected Layer	1024 Units, Dropout = 0.5
Activation Function	ReLU
Fully Connected Layer	10 Units
Output Layer	Softmax

The loss function used in this study is categorical cross-entropy:

$$L = -\sum_{i=1}^{C} y_i log(p_i) \tag{6}$$

Where: C is the total number of categories, y_i is the encoding of the real label,will the real category is 1, and the rest are 0, p_i is The category probability predicted by the model, output through Softmax, $\sum p_i = 1$.

The optimization algorithm is Adam (Adaptive Moment Estimation), and the performance metric is accuracy.

Adam is an optimization algorithm that combines Momentum and adaptive learning rate, and is widely used in the training of deep learning models. It significantly improves the efficiency and stability of gradient descent by dynamically adjusting the learning rate of each parameter, and is particularly suitable for dealing with problems of sparse gradients or non-stationary objective functions (such as CNN, RNN, etc.).

The momentum part is similar to the traditional momentum method, recording the exponential moving average of the gradient to accelerate convergence and reduce oscillation.The adaptive learning rate part calculates the exponential moving average of the square of the gradient and assigns dynamic learning rates to different parameters to alleviate the problems caused by the differences in the magnitude of the gradient.

This research utilizes the MNIST dataset, which contains 70,000 samples—60,000 for training and 10,000 for testing—covering digits 0 through 9. The number of samples in each category is approximately balanced. Each image has dimensions of 28×28 pixels and is paired with a corresponding label indicating its true value. All images are centered and uniformly scaled.

During training, the dataset images were preprocessed and converted into arrays stored in CSV files. The batch size was set to 50 samples per group, and the model was trained for one epoch. Both the training and validation sets were divided into groups of 50 samples. The final model achieved an accuracy of approximately 0.9875. Some sample data from the dataset are shown below:

DOI: 10.54254/2755-2721/2025.PO24694

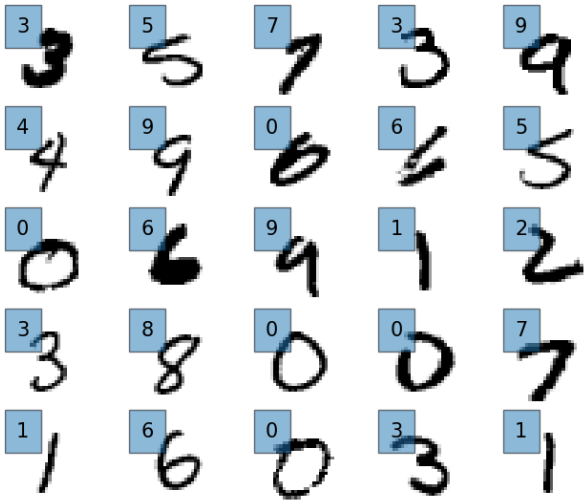


Figure 2: Output sample

Selected twenty-five samples randomly from the test set as examples. It can be seen that in the output samples, the predicted values of the pictures are displayed in the upper left corner of the handwritten digits.

4. Discussion

This study has room for improvement. The research uses a large number of samples for handwritten digit recognition, resulting in significant computational demands. Potential enhancements include reducing the learning rate and increasing the dropout ratio to improve computational efficiency. The neural network in this study utilizes fixed-size 28×28 image samples, which limits its ability to recognize inputs of varying dimensions. Incorporating a rescaling module to standardize image sizes would enable the model to recognize dynamically sized images.

When addressing more complex mathematical expression recognition tasks, the model requires further optimization. On the one hand, for symbol recognition tasks, it is necessary to increase the feature dimensions at the output layer. On the other hand, for recognizing long mathematical expressions, additional image segmentation algorithms should be introduced upstream. These modifications would better reflect real-world application scenarios. However, such changes may affect the model's accuracy. Expanding the recognition categories could result in reduced accuracy rates.

5. Conclusion

This study employed the TensorFlow framework to construct a convolutional neural network for handwritten digit recognition, using experiments conducted on the MNIST dataset. The CNN model achieved significantly higher accuracy than traditional SVMs or fully connected networks, demonstrating strong generalization capabilities for digit recognition. Incorporating components such as convolutional layers, pooling layers, ReLU activation functions, and Dropout, the model achieved an accuracy of 98.75% on the MNIST dataset. While the model performs excellently in

DOI: 10.54254/2755-2721/2025.PO24694

single-digit recognition, its generalization performance in more complex scenarios requires further evaluation. The next step is to combine this model with other modules to enhance its generalization ability, so as to achieve application standards that are more in line with the actual situation.

References

[1] Li, S. F., & Gao, F. Q. (2017). Handwritten numeral recognition based on convolutional neural network. Journal of Zhejiang Sci-Tech University (Natural Sciences), 37(3), 6.
[2] Huang, R., Lu, S. M., & Wu, Y. L. (2018). Handwriting digit recognition and application based on TensorFlow deep learning. Application of Electronic Technique, 44(10), 5.
[3] Tang, J. B., Li, W. J., Zhao, B., & Xi, L. P. (2022). Research on handwritten digits recognition method based on convolutional neural network. Electronic Design Engineering, 30(21), 189–193. https://doi.org/10.14022/j.issn1674-6236.2022.21.040
[4] Ji, X. (2023). A research and application of handwritten digit recognition based on convolutional neural networks. Information & Computer, 35(12), 169–171, 183.
[5] Dutta, K. (2018). Improving CNN-RNN hybrid networks for handwriting recognition. In 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR).
[6] Yang, G., He, D. G., & Dai, L. Z. (n.d.). Improved handwritten digit recognition based on CNN and PSO-SVM. Journal of East China Jiaotong University, 37(4), 7.
[7] Deepa, S. (2023). Handwritten digit recognition using multilayer deep convolutional neural network. In 2023 Third International Conference on Smart Technologies, Communication and Robotics (STCR).
[8] Srivastav, S. (2023). Handwritten digit recognition using fine-tuned convolutional neural network model. In 2023 Third International Conference on Smart Technologies, Communication and Robotics (STCR).
[9] Zhang, M. Y., Ye, H. C., Yuan, X. F., & Chen, H. Y. (2024). Cross-domain author-adaptive handwritten recognition based on CNN-ISVM. Computer Technology and Development, 34(12), 187–193. https://doi.org/10.20165/j.cnki.ISSN1673-629X.2024.0235