Real-Time Hand Written Digit Recognition

Akshay Bajpai and Patrick Huynh
Department of $Computer Science$
University at Buffalo
Buffalo, NY 142603
{akshayus;tienhuyn}@buffalo.edu

Abstract

A basic machine learning problem, handwritten digit recognition has numerous applications in fields including automated form processing, banking, and education. In this research, we categorize handwritten digits with reliability using a **Convolutional Neural Network (CNN)**. The popular **MNIST dataset**, which comprises 10,000 test and 60,000 training samples of 28x28 grayscale images, is used to train and assess the model. Our approach includes regularization dropout, data augmentation (rotation and translation), and data pretreatment (normalization and reshaping) to prevent overfitting. The trained model has been adapted to handle **real-time user-drawn inputs** via an interactive graphical interface and shows good accuracy on the MNIST test set. The resilience and applicability of our method for dynamic applications are demonstrated by evaluating the model using common metrics including accuracy, confusion matrices, and qualitative feedback on user inputs.

1 Introduction

With the technological application of most AI models in our devices, handwritten recognition is one of the key ideas for transferring what we draw and write into digital data for easier access in the digital world. The standard neural network could achieve the most simple handwritten digit recognition. Still, such an approach could not produce our expected highly accurate result since this approach would only take inputs and process them as vectors, neglecting the links between specific elements of the handwritten digits, letters, and words. Therefore, we tried to apply CNN to this application, leading to a more precise outcome. CNN is used wisely to capture images, and compared to traditional neural networks, it could process images of more extensive data. With more advanced features than the conventional neural network, we chose CNN to recognize handwritten words' curvature, texture, and colors. Overall, this approach will enhance the accuracy of outcomes and feasibly process more significant data as input.

2 Related Works

In machine learning, handwritten digit recognition has been thoroughly investigated, and several methods have been developed to tackle the issue utilizing datasets such as MNIST. LeCun et al.'s LeCun et al. [1998] introduction of the MNIST dataset and a neural network model for digit classification based on backpropagation was one of the pioneering studies in this field. Due to the inefficiency of fully connected layers for spatial data, this work had limits even though it set the standard for neural network applications in image recognition.

As noted by Cortes and Vapnik Cortes and Vapnik [1995], Support Vector Machines (SVMs) have also been utilized extensively for digit recognition. SVMs performed well on smaller datasets and

excelled at binary classification tasks, but they were unable to scale for large-scale image datasets such as MNIST. Deep neural networks with GPU acceleration were used in later developments, as those by Ciresan et al. Ciresan et al. [2012], which achieved state-of-the-art accuracy on MNIST. Their research showed how deep learning may use computing power to outperform conventional machine learning techniques.

In order to increase the model's resilience to handwriting variances, Simard et al. Simard et al. [2003] stressed the significance of data augmentation approaches like translation and rotation. By randomly deactivating neurons during training, dropout—a technique first presented by Srivastava et al. Srivastava et al. [2014]—was a crucial step in reducing overfitting in deep neural networks.

By utilizing a convolutional neural network (CNN) architecture that is designed for accuracy and efficiency, our study builds upon previous developments. To improve performance and resilience, we use dropout layers and data augmentation in contrast to LeCun's conventional neural network or SVM-based approaches. Furthermore, our project incorporates real-time user-drawn digit prediction, enabling interactive applications, whereas previous works, such as those by Ciresan et al. Ciresan et al. [2012], concentrated on offline digit recognition. This real-time capability demonstrates how conventional approaches may be practically extended to dynamic use cases.

3 Data

The dataset we used is the **MNIST Dataset**. Each image's type is visual, grayscale, and represented as a 28×28 matrix, where pixel intensity is between 0(black) and 255(white). The dataset has 60,000 training data images and 10,000 testing data images.

The dataset is generally used for handwritten digit recognition work using neural networks. Originally, we were inspired by the work of GeeksforGeeks GeeksforGeeks [2024] and used the same sample dataset to improve on the work of using CNN instead of a traditional neural network Community [2024].

Ultimately, in our project, we imported directly from TensorFlow/Keras.

3.1 Preprocessing Steps

We have used preprocessing step in our project to achieve the expected result.

1. **Normalization**: We scale pixel to fit the value between 0 and 1 for training.

2. **Reshaping**: We reshaped the input data to match the format for inputting into our CNN model

```
X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)
```

3. **One-Hot Encoding**: We convert labels of data into categories to be activated by softmax for later work.

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

3.2 Dataset Example

Figure 1 shows examples of MNIST digit images.

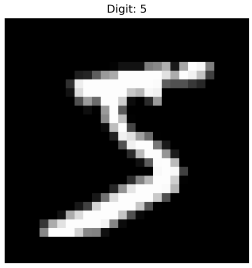

Figure 1: Example of MNIST handwritten digits represented in grayscale form.

4 Methods

We used a **Convolutional Neural Network (CNN)** to address the handwritten digit recognition challenge because of its applicability to image-based tasks. By using convolutional layers to extract features like edges, curves, and forms, CNNs are excellent at capturing spatial relationships in visual data. CNNs are the best option for digit recognition because of the automated feature extraction that eliminates the need for manual engineering.

Our method's efficacy rests in its capacity to strike a compromise between computational efficiency and accuracy. Through a sequence of convolutional and pooling layers, the CNN model reduces dimensionality while preserving crucial information in images. We use strategies like **dropout regularization**, which avoids overfitting by randomly deactivating neurons during training, to make sure the model generalizes well to unseen data. To improve the training dataset's diversity and strengthen the model's resistance to handwriting variances, data augmentation techniques such as image translation and rotation are also used.

Other approaches were taken into consideration when constructing the solution. Despite their ease of implementation, traditional fully connected neural networks ignore the spatial correlations necessary for precise image recognition and treat each pixel independently. Similarly, **Support Vector Machines (SVMs)** have demonstrated excellent performance on smaller datasets, but their computational inefficiency makes them difficult to scale for larger datasets like MNIST. Even more accuracy would be possible with more sophisticated architectures like **ResNet** or **VGG**, but these models are computationally costly and less appropriate for real-time, lightweight applications. In order to attain high performance while preserving efficiency, a comparatively straightforward yet optimized CNN architecture was selected.

Figure 2 provides an illustration of the CNN architecture used in this study.

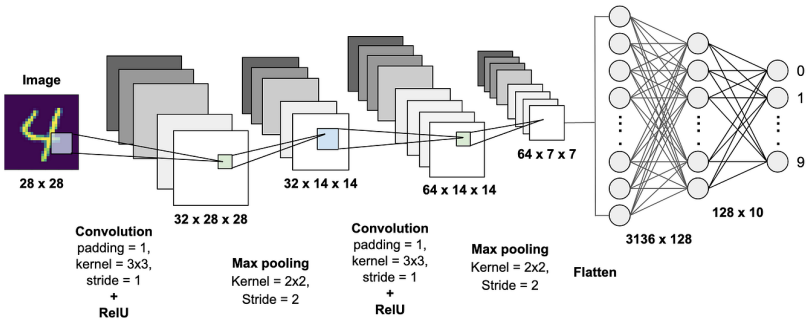

Figure 2: The architecture of the Convolutional Neural Network (CNN) used for handwritten digit recognition.

Source: Adapted from Patel Patel [2020].

Our strategy includes a few crucial steps. First, preprocessing is done on the **MNIST dataset**, which includes 10,000 test samples and 60,000 training samples of 28x28 grayscale images. To guarantee steady and quicker convergence during training, the images are normalized to pixel values in the [0,1] range. Additionally, the input images are resized to 28x28x1 in order to conform to the CNN's required format. The **Adam optimizer**, which effectively modifies weights to minimize the categorical cross-entropy loss function, is then used to train the CNN. To maximize model performance, hyperparameters including learning rate, batch size, and dropout rate were adjusted.

5 Experiments and Results

5.1 Comparison with Previous Published Project

We compared our CNN approach with a traditional neural network and evaluated their performance on the MNIST dataset. Source of traditional neural network project: GeeksforGeeks: Handwritten Digit Recognition using Neural Network Tutorial: Handwritten Digit Recognition using Neural Network.

Table 1: Comparison of Traditional NN and CNN Models

Criteria	**Traditional NN**	**CNN**
Accuracy (%)	99.53	98.89
Model Complexity	Simple	Complex
Training Time	Faster	Slower
Scalability	Limited	High

5.2 Ablation Study

We removed certain layers and observed the effects to evaluate the impact of our CNN components

Table 2: Ablation Study Results for CNN

Experiment	**Accuracy (%)**
Baseline CNN (All Layers)	98.89
Without Max-Pooling	97.12
Without Dropout Regularization	96.80
Without Second Convolutional Layer	96.50

5.3 Hyperparameter Tuning

We experimented with hyperparameters including learning rate, batch size, and kernel size, in order to come up with the optimal configuration.

Table 3: Hyperparameter Tuning Results

Hyperparameter	**Value**	**Accuracy (%)**
Learning Rate	0.01	97.50
Learning Rate	0.001	98.89
Batch Size	32	98.50
Batch Size	64	98.89
Kernel Size	3×3	98.89
Kernel Size	5×5	98.20

5.4 Visualization of Model Performance

In order to better understand how the CNN analyses the data and makes decisions, we employed visualization approaches.

- **Feature Maps**: The first convolutional layer's outputs demonstrate that the CNN is able to identify textures and edges in the input images.
- **Grad-CAM**: The regions of input photos that made the most contributions to the model's predictions are highlighted using Grad-CAM (Gradient-weighted Class Activation Mapping).

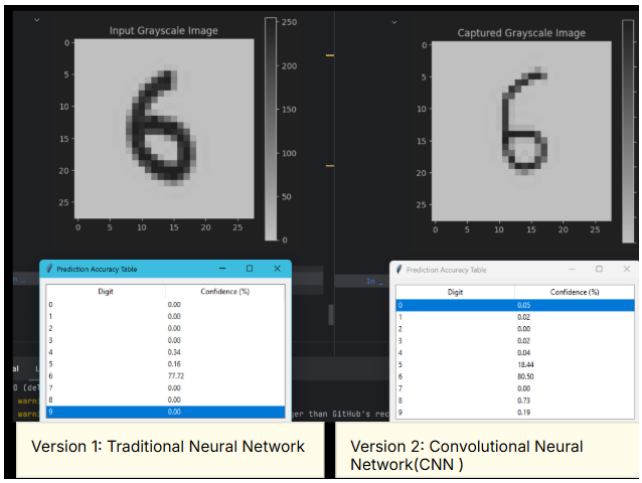

Figure 3: Grad-CAM Visualization for Digit '6'.

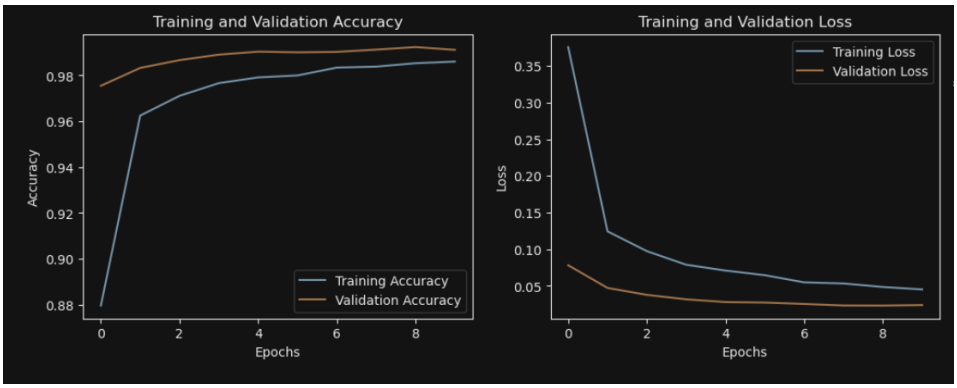

Figure 4: Training and Validation Accuracy (Left) and Training and Validation Loss (Right) over Epochs for CNN Model.

5.5 Failure Modes

The CNN model showed several possibilities for failure despite its high accuracy. These included poor input photos and uncertain numbers.

Table 4: Failure Cases of CNN Model

True Label	Predicted Label
4	9
7	1

5.6 Evaluation Metrics

To evaluate model performance, we applied the following evaluation metrics:

- **Accuracy**:Total proportion of accurate predictions.

- **Loss**: Prediction error is evaluated using cross-entropy loss.
- **Confusion Matrix**: Misclassifications in all classes (digits 0–9) were analyzed.0–9).

6 Conclusion and future work

This research successfully implemented **Convolutional Neural Networks (CNNs)**, particularly for the identification of handwritten digits. Using the **MNIST dataset**, our model demonstrated strong performance, achieving significant accuracy. Additionally, we extended the model to handle real-time digit recognition through an interactive graphical user interface, enabling users to draw digits dynamically. A combination of data preparation techniques, including normalization and reshaping, along with **data augmentation** methods such as rotation and translation, and **dropout regularization** ensured robustness against overfitting. These steps enabled the model to perform effectively on both test data and user-generated inputs. This work clearly demonstrates the ability of CNNs to classify digits with minimal computational costs while meeting the demands of real-time applications.

Although our model achieved positive results, certain areas remain uninvestigated. Noisy or low-quality user-drawn inputs can still lead to incorrect classifications. To address this, the preprocessing pipeline could be enhanced to account for handwriting variations such as faint strokes and overlapping numerals. Another limitation lies in the model's current reliance on single-digit classification; it cannot recognize multiple digits in a single input. Extending the model to handle multi-digit recognition or even handwritten mathematical expressions would significantly broaden its applicability.

Future improvements should focus on making the model feasible for deployment on **mobile devices** or **edge platforms**. Lightweight architectures such as **TinyCNN** or **MobileNet** could reduce computational requirements while maintaining good accuracy, enabling real-time handwritten digit recognition on devices with limited resources. Additionally, the model's performance on user-drawn inputs could be further improved by incorporating techniques such as **transfer learning**, particularly in scenarios with high real-world variability. Another promising direction would be integrating the system into practical applications, such as assistive tools for individuals with disabilities, educational resources for children, or automated form digitization.

In summary, this work establishes a strong foundation for real-time handwritten digit recognition using CNNs. By addressing the identified limitations and exploring future extensions, the system can be further enhanced to meet diverse practical needs and deliver greater societal impact.

References

Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649, 2012.

Kaggle Community. Handwritten digit recognition using neural network, 2024. URL `https://www.kaggle.com/datasets/subho117/handwritten-digit-recognition-using-neural-network/data`. Accessed: June 17, 2024.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

GeeksforGeeks. Handwritten digit recognition using neural network, 2024. URL `https://www.geeksforgeeks.org/handwritten-digit-recognition-using-neural-network/`. Accessed: June 17, 2024.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Krut Patel. Mnist handwritten digits classification using a convolutional neural network (cnn), 2020. URL `https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9`. Published on Towards Data Science, Accessed: June 17, 2024.

Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 958–962. IEEE, 2003.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Acknowledgments

We acknowledge the fact that attempt to complete this project, other tools and resources were used. More specifically, we used Grammarly and ChatGPT to improve our academic writing vocabulary, structure unstructured claims, and fix and correct grammatical faults. Furthermore, ChatGPT was used to provide advice on the coding parts of our project, including Convolutional Neural Network (CNN) theories, debugging, and improving our model's learning and evaluation.