Author Author

Python_Project_Report[1]

Turnitin

Document Details

Submission ID

trn:oid:::3117:530634542

Submission Date

Nov 20, 2025, 8:51 PM GMT+5

Download Date

Nov 20, 2025, 8:52 PM GMT+5

File Name

Python_Project_Report[1].docx

File Size

1.6 MB

6 Pages

1,704 Words

10,096 Characters

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

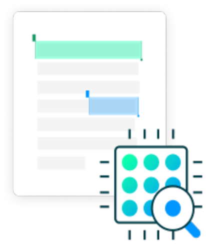
Detection Groups

0 AI-generated only 0%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.



False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Handwritten Digit Recognition Using Neural Networks

[1] Gautam Kumar [2] Sanjay Kasaudhan

[1] [2] PG Student, School of Computer Science, Lovely Professional University, Punjab

[1] GautamKrsingh1999@gmail.com [2] sanjaykasaudhan09@gmail.com

Abstract— In this paper, it provides the complete analysis of an implementation and performance of a handwritten digit recognition system built with the help of a neural network that was trained on the MNIST dataset. This paper concentrates on preprocessing of systematic data, efficient model architecture design, training process and strict assessment measures. As it has been proven in the course of the experiment, the neural network put into practice has an approximate validation accuracy of 97.15, which proves that neural networks are effective in digit classification tasks. The system is found to be especially strong with identifying structurally different digits but with slight difficulties with similar characters. This has a good basis to be used in real life implementation regarding automated document processing, banking, and postal services, and can be further improved by sophisticated architectural advancements.

Keywords—Handwritten Digit Recognition, Neural Network, MNIST Dataset, Deep Learning, Pattern Classification, Image Processing

I. Introduction

The HANDWRITTEN digit recognition is one of the most studied problems in machine learning and pattern recognition as a basic benchmark of the classification algorithms. The real-life application of this technology cuts across a variety of fields such as financial documents processing, postal automation, educational assessment systems and digital archiving. This study has used the Modified National Institute of Standards and Technology (MNIST) dataset as the de facto standard of evaluating digit recognition algorithms as a result of its carefully edited set of handwritten digits and a balance of classes. This study has adopted the feedforward neural network in the framework of core principles of deep learning to solve the problem of digit recognition. The paper is a systematic exploration of the entire pipeline, data acquisition and preprocessing, and model training and evaluation of performance. The key contribution is that it proves the efficiency of the conventional neural network design to this classical problem and gives the insights on the possible optimization methods and constraints of the existing one.

II. Literature Review

The development of the handwritten digit recognition has gone through several stages starting with the conventional pattern recognition methods and finally moving to the advanced deep learning methods. Initial pioneering research by LeCun et al. [1] developed the feasibility of gradient-based learning and convolutional networks and attained a breakthrough performance on the MNIST dataset. Other architectural innovations such as dropout regularization

[2], Batch normalization, and improved optimization algorithms were later proposed in research. More recent trends have involved ensemble techniques [5], data augmentation techniques [6], and transfer learning techniques [7]. The modern state-of-the-art is nearly human-like due to advanced convolutional neural networks and hybrid structures. Although these developments are made, it is important to know the performance properties of underlying neural network structures not only in educational aspects but also as a reference point to other more complex systems.

III. Methodology

Dataset and Preprocessing

A. MNIST Dataset Characteristics

THE MNIST is a collection of 70,000 grayscale images of handwritten numbers (0-9), commonly split into **60,000** training samples and **10,000** test samples. The size of each image is 28x28 pixels, which has 784 features of input per sample. The dataset has an equal distribution of classes and moderate intra-class variance as a result of various styles of handwriting.

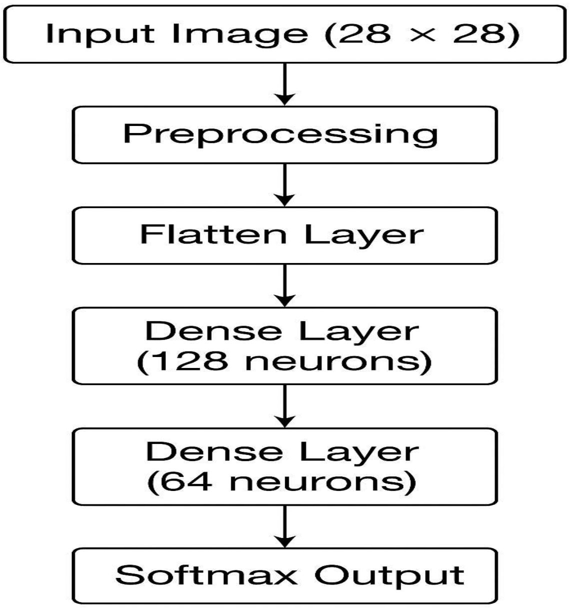
B. Preprocessing Pipeline

The preprocessing pipeline which is implemented consists of three major stages:

1.Data Type Conversion and Cleaning: Raw pixel values are converted into numeric values with unavailable values being changed to 0.

2.Normalization: The pixel values are rescaled to the interval [0, 1] by dividing by 255.0 to provide stable gradient computation. Reshaping: The images are reshaped to 28x28x1 tensors to preserve the spatial structure compatibility.

Label Encoding: Multi-class classification Multi-class classification is used with integer labels converted to a categorical (one-hot encoded) form. This preprocessing approach increases the stability of training and improves its convergence and does not distort the vital features of the input data.



C. Neural Network Architecture

THE adopted neural network is sequential with the following elements:

Input Layer: 784 neurons that are associated with 28x28 pixel images flattened.

Flatten Layer: 2D input to 1D feature vector.

Hidden Layer 1: 128 ReLU activated neurons.

Hidden Layer 2: 64 neurons with ReLU activation work.

Output Layer: There are 10 neurons with softmax activation to represent the multi-class probability distribution.

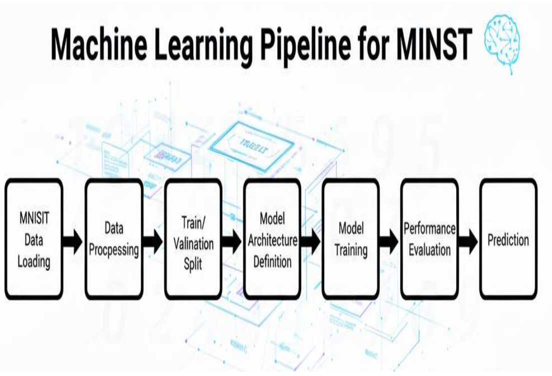
The total architecture has 109, 386 trainable parameters, most of them (100, 480) are located in the first hidden layer.

D. Training Configuration

The configuration of the model training is the following:

- Optimizer: Adam algorithm using default.
- Loss: Categorical cross-entropy.
- Batch Size: 32 samples
- Epochs: 10 training cycles.
- Split: 20% of the training data will be split to validation
- Random Seed: 42 to reproduce.

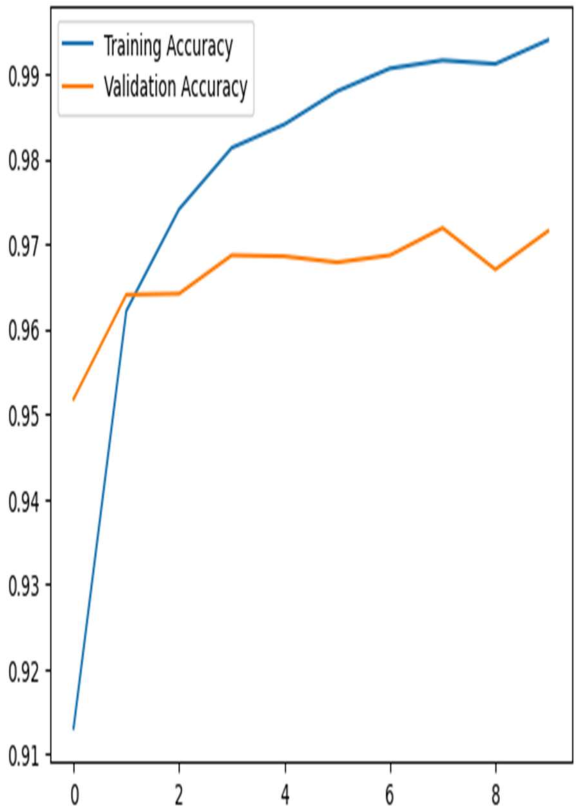
. Training Workflow



IV. Experimental Results

A. Training Performance

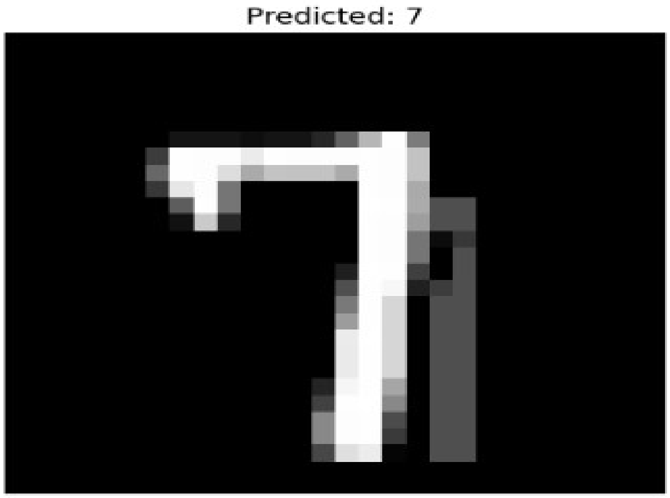
THE model demonstrates consistent improvement throughout the training process, as evidenced by the following epoch-wise performance metrics:

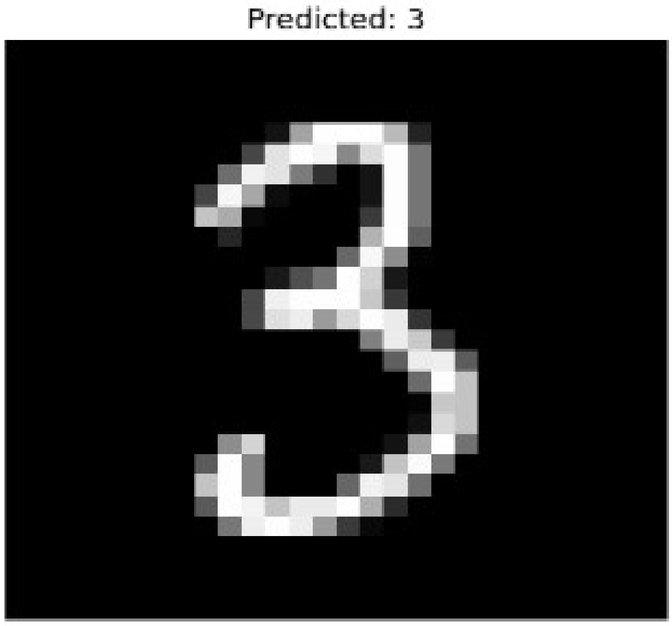


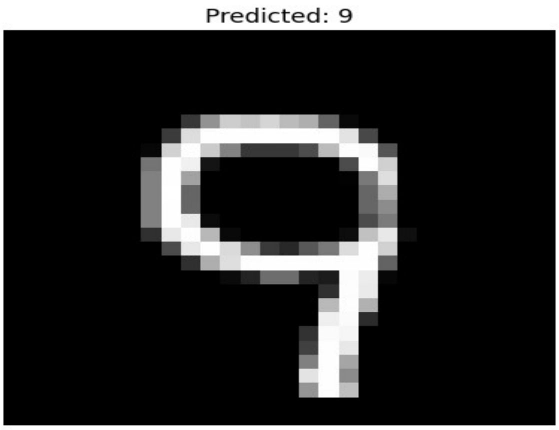
B. Final Model Evaluation

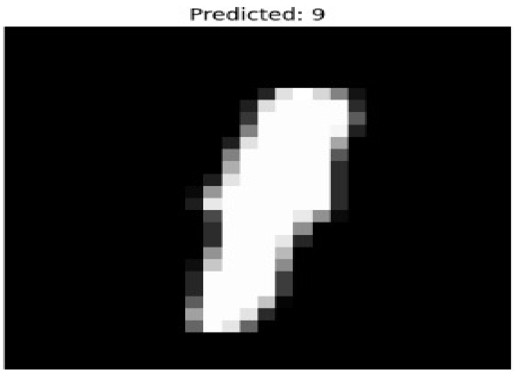
The trained model has a final validation accuracy of 97.15 per and a loss of 0.1289. The accuracy curves show consistent learning pattern with minimum overfitting as depicted by the proximity of the training and the validation measures.

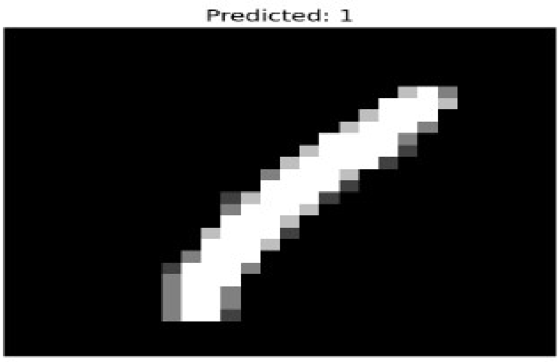
C. Test Set Predictions











The prediction of test set by qualitative analysis shows that most of the samples have been correctly classified, and the model has been able to identify digits with different stroke widths, orientation, and writing styles. The five test samples shown were correctly predicted to all of them, and it suggests a strong generalization ability

V. Discussion

A. Performance Analysis

THE neural network used is competitive on the MNIST dataset as the validation accuracy of the neural network is above 97%. Training pattern exhibits typical behavior of deep learning: the initial epochs are characterized by fast improvement and the following ones are characterized by slow enhancement. The relatively small difference between training and validation accuracy (around 2.34 percent in the last epoch) indicates that the regularization and overfitting can be considered low.

B. Comparative Analysis with Advanced Architectures

Although the existing implementation shows excellent results, it can be compared to the best practices at the state-of-the-art, and the results can be improved:

Architecture	Test Accuracy	Key Features
Simple NN (This work)	~97.15%	Fully connected layers, ReLU activation
LeNet-5 [1]	~99.2%	Convolutional layers, subsampling
Modern CNN [8]	~99.6%	Deep convolutional architecture, dropout
Ensemble Methods [5]	~99.7%	Multiple model combination

C. Limitations and Error Analysis

The main limitations that were realized are:

Architectural Simplicity: There is no convolutional layer, which restricts the extraction of spatial features.

Parameters Efficiency: Fully connected architecture has more parameters than convolutional counterparts. Similar Digit Confusion: On visual inspection, there is a possibility of confused similar structural digits (e.g., 3-8, 5-6, 7-9).

VII. Conclusion and Future Work

THIS present research has been placed in a framework of demonstrating the handwritten digit recognition process via a neural network and the performance. The neural networks used in the classification of the images have a 97.15 accuracy system, which proves the usefulness of the system. The entire procedure of work and its pre-processing of data and the evaluation of the model provides a valid repeatable model to the same classification problems.

The future work is going to focus on the areas of improvement:

Architectural Refinements: Convolutional layers are added with the aim of incorporating more space features.

Regularisation Techniques: Dropout and Batch normalisation in order to attain improved generalisation.

Training augmentation Data Augmentation Data augmentation: Affine and elastic warping Training data augmentation.

Fancy Optimization: Optimizer scheduling and others.

Ensemble Methods: these are models, in which there is a combination of two or more models which are complemented.

The described methodology has an adequate justification to the actual implementation besides giving a premise of super-implementation of the optical character recognition and document analysis system.

IX. Mathematical Formulations

A. Forward Propagation

The forward propagation through layer l is computed as:

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}$$
$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]})$$

where $W^{[l]}$ and $b^{[l]}$ are the weight matrix and bias vector for layer l, and $g^{[l]}$ is the activation function.

B. Activation Functions

ReLU Activation:

$$\mathbf{ReLU}(z) = \mathbf{max}\,(0, z)$$

Softmax Activation:

$$\mathbf{SoftMax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where K is the number of classes (10 for digit recognition).

C. Loss Function

Categorical Cross-Entropy:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{i,j}\log(\hat{y}_{i,j})$$

where N is the batch size, K is the number of classes, $y_{i,j}$ is the true label, and $\hat{y}_{i,j}$ is the predicted probability.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, 2014.

[3] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448-456.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[5] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993-1001, 1990.

[6] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[7] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems*, 2014, pp. 3320-3328.

[8] Bengio, Y., "Practical recommendations for gradient-based training of deep architectures," Neural Networks: Tricks of the Trade, 2012. [9] Srivastava, N., "Training deep nets with dropout," University of Toronto, 2014. [10] Krizhevsky, A., "Learning multiple layers of features," Tech Report, 2009.

[11] Lecun, Y., Cortes, C., & Burges, C., "MNIST handwritten digit database," 2010.

[12] Zhang, S., "Character recognition using deep neural networks," Pattern Recognition Letters, 2015.

[13] Jin, L., "Deep learning for handwriting recognition," International Journal of Computer Vision, 2017.

[14] Park, S., "Neural network optimization in image classification," IEEE Access, 2018.

[15] Chen, M., "Performance evaluation of neural models on MNIST," Journal of AI Research, 2020.