Indonesian Journal of Electrical Engineering and Computer Science
Vol. 44, No. 1, October 2026, pp. 283~292
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v44.i1.pp283-292

A compact lightweight convolutional neural network for handwritten digit recognition

Shehla Afzal[1], Muhammad Talha Farooqi[1], Irfanud Din[2], Tayyaba Anees[1]
[1]Department of Software Engineering, School of Systems and Technology, Umiversity of Management and Technology (UMT), Lahore, Pakistan
[2]Department of Computer Science, New Uzbekistan University (NewUU), Tashkent, Uzbekistan

Article Info

Article history:

Received Jul 31, 2026
Revised Aug 28, 2026
Accepted Sep 9, 2026

Keywords:

Convolutional neural network
Handwritten digit recognition
Hyperparameter optimization
Deep learning
MNIST

ABSTRACT

Handwritten digit recognition (HDR) remains a fundamental research area in computer vision and pattern recognition due to its extensive applications in optical character recognition (OCR), postal mail sorting, bank cheque verification, document digitization, and intelligent document processing. Although the modified national institute of standards and technology (MNIST) dataset has become a well-established benchmark, designing compact convolutional neural network (CNN) architectures that achieve high recognition accuracy with rapid convergence and low computational requirements continues to be an active research challenge. This paper presents a compact CNN architecture comprising three convolutional layers followed by two fully connected layers for handwritten digit classification. In addition, the study investigates the influence of key training hyperparameters, including batch size, kernel size, and optimizer, on model convergence and classification performance. Experimental evaluation on the MNIST dataset demonstrates that the proposed model achieves 99.37% training accuracy and 99.15% testing accuracy after 10 training epochs, with the highest validation accuracy of 99.20% obtained at the sixth epoch. The model contains 333,450 trainable parameters, has an FP32 model size of approximately 1.27 MB, and requires approximately 73.23 million floating-point operations (FLOPs) per inference. The confusion matrix indicates that the most frequent misclassifications involve visually similar digit pairs, particularly 4–9, 6–5, and 3–5. The findings demonstrate that careful CNN architecture design and empirical hyperparameter analysis can provide accurate and computationally compact HDR, with potential suitability for resource-constrained applications.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Shehla Afzal
Department of Software Engineering, School of Systems and Technology
University of Management and Technology (UMT)
Lahore, Pakistan
Email: shehla.afzal@umt.edu.pk

1. INTRODUCTION

Handwritten digit recognition (HDR) is one of the most fundamental and extensively studied problems in computer vision, pattern recognition, and machine learning because of its broad range of real-world applications, including postal mail sorting, bank cheque processing, optical character recognition (OCR), document digitization, automated form processing, and intelligent archival systems [1]–[3]. Accurate handwritten digit classification is an essential component of numerous industrial and commercial applications where reliable recognition of handwritten numerical information is required. Consequently, HDR continues

to serve as a standard benchmark for evaluating emerging deep learning algorithms and image classification techniques.

The rapid advancement of deep learning has significantly transformed HDR. Earlier approaches primarily relied on handcrafted feature extraction methods combined with traditional machine learning classifiers such as support vector machines (SVM), decision trees, k-Nearest Neighbors (k-NN), and multi-layer perceptrons (MLPs). Although these methods produced satisfactory results, their performance largely depended on manually engineered features and domain expertise. The introduction of convolutional neural networks (CNNs), particularly the pioneering LeNet architecture proposed by LeCun *et al.*, revolutionized HDR by enabling automatic hierarchical feature learning directly from image pixels [4], [5]. Since then, CNN-based models have consistently achieved state-of-the-art performance on the MNIST benchmark while requiring minimal feature engineering.

The modified national institute of standards and technology (MNIST) dataset remains one of the most widely used benchmark datasets for evaluating HDR algorithms [6]-[11]. Despite classification accuracy approaching saturation on MNIST, recent research has shifted its focus toward designing lightweight and computationally efficient CNN architectures that achieve high recognition accuracy while reducing model complexity, computational cost, and training time [12]–[17]. Such lightweight models are particularly attractive for deployment in embedded systems, mobile devices, and other resource-constrained environments where computational resources are limited.

Recent studies have also emphasized the importance of optimization strategies beyond network architecture alone. Hyperparameters such as learning rate, batch size, kernel size, activation function, optimizer selection, and normalization techniques have been shown to significantly influence convergence speed, model stability, and generalization performance [17]–[21]. Furthermore, comparative studies involving lightweight CNNs, attention-based networks, and vision transformer (ViT) architectures indicate that carefully optimized CNN models continue to offer an excellent balance between computational efficiency and recognition accuracy for HDR tasks [19], [22]-[26].

Although numerous CNN-based approaches have reported recognition accuracies exceeding 99% on the MNIST dataset, several limitations remain. Many studies focus primarily on maximizing classification accuracy while providing limited investigation into the combined effects of multiple hyperparameters on convergence behaviour. Moreover, several high-performing models employ increasingly deeper and more computationally intensive architectures, making them less suitable for practical deployment on low-resource hardware [17], [19], [20].

To address these challenges, this paper proposes a compact lightweight CNN comprising three convolutional layers followed by two fully connected layers for HDR. In contrast to many existing studies, this work not only evaluates classification accuracy but also investigates the influence of batch size, kernel size, and optimizer on convergence and overall model performance. Extensive experiments conducted on the MNIST dataset demonstrate that the proposed architecture achieves 99.37% training accuracy and 99.15% testing accuracy within only 10 training epochs, while the highest validation accuracy of 99.20% is obtained at epoch 6. In addition, detailed confusion matrix analysis and class-wise performance evaluation are performed to identify the remaining challenging digit pairs that limit further improvement.

The main contributions of this research are summarized as follows:

- A compact CNN architecture consisting of three convolutional layers and two fully connected layers for efficient HDR.
- A systematic empirical evaluation of the influence of batch size, kernel size, and optimizer on convergence behaviour and classification performance.
- Comprehensive performance analysis through confusion matrix evaluation, class-wise precision, recall, F1-score, and comparison with recently published CNN-based HDR methods.
- Competitive classification performance, achieving 99.37% training accuracy, 99.15% testing accuracy, and 99.20% validation accuracy with rapid convergence in only 10 training epochs.

The remainder of this paper is organized as follows. Section 2 reviews recent literature related to HDR. Section 3 presents the proposed CNN architecture and methodology. Section 4 discusses the experimental setup and performance evaluation. Section 5 presents the limitations and future research directions, while section 6, concludes the paper.

2. METHOD

The proposed pipeline follows the standard CNN paradigm—feature extraction by stacked convolution-activation-pooling blocks, classification by fully connected layers, and parameter updates by back-propagation—but is engineered for fast convergence on MNIST. The model is implemented in Keras with a TensorFlow backend.

2.1. Network architecture

The proposed network, illustrated in Figure 1, consists of three convolutional blocks for hierarchical feature extraction followed by two dense layers for classification. The components are:

- Input layer. Each input is a 28×28×1 grayscale tensor (single channel, pixel values normalized to [0, 1]).
- Conv block 1. Conv2D with 64 filters of size 3×3, stride 1, ReLU activation, followed by MaxPooling2D with a 2×2 pool.
- Conv block 2. Conv2D with 256 filters of size 3×3, ReLU activation, followed by MaxPooling2D 2×2.
- Conv block 3. Conv2D with 64 filters of size 3×3, ReLU activation, followed by MaxPooling2D 2×2.
- Flatten layer. The 2D feature maps from the last pooling layer are flattened to a 1D vector.
- Dense layer. A fully connected layer with 64 ReLU units, with dropout (rate = 0.5) for regularization.
- Output layer. A 10-unit dense layer with softmax activation, producing the class-probability distribution over digits 0–9.

ReLU is preferred over sigmoid/tanh because it avoids the saturation/vanishing-gradient problem in shallow stacks; max-pooling is preferred over average-pooling because it preserves the strongest activations—typically the strokes that distinguish one digit from another. The 64–256–64 filter progression was retained as the configuration identified for the proposed model. The larger intermediate convolutional layer provides increased representational capacity for learning diverse stroke-level and shape-related features, while reducing the number of filters in the third block limits the subsequent feature-map dimensionality before the fully connected layers. Thus, the architecture balances intermediate feature extraction capacity with a relatively compact overall parameter count."

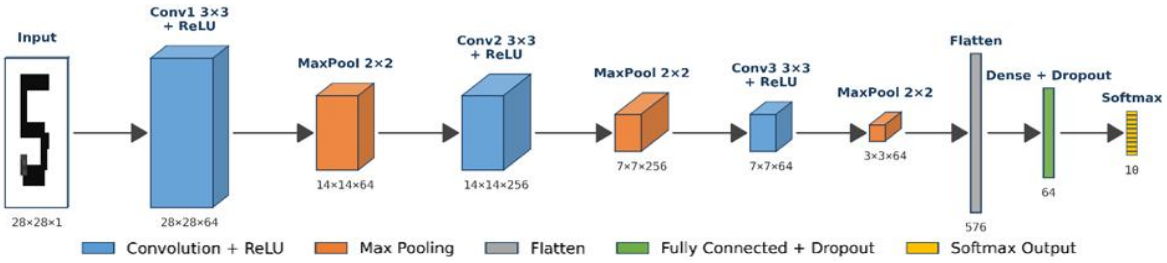

Figure 1. Block diagram of the proposed three-Conv / two-Dense CNN. Input is a 28×28×1 grayscale image; each Conv block is followed by ReLU activation and 2×2 max-pooling; flattening is followed by a 64-unit dense layer with dropout, and a 10-unit softmax output

2.2. Hyperparameter optimization strategy

Three hyperparameter settings were investigated for their influence on convergence and classification performance on MNIST: batch size, convolutional kernel size, and optimizer. We swept each in a one-factor-at-a-time fashion, holding the others at sensible defaults, and selected the configuration that minimized validation loss after the first five epochs.

- Batch size: Very small batches (e.g., 8–16) introduce gradient oscillations and slow convergence; very large batches (e.g., 256+) tend to converge to sharp local minima with poorer generalization. Based on the sensitivity experiments, batch size = 50 was retained for the final model as a competitive configuration that provided stable convergence while maintaining the adopted training setup.
- Kernel size: All three convolutional layers use 3×3 kernels. The experiments compared 3×3 and 5×5 kernels under the same training conditions. Although the 5×5 configuration achieved slightly higher performance in the five-epoch sensitivity experiment, 3×3 kernels were retained in the final architecture because they provide a more compact convolutional design and are applied consistently across all three convolutional layers.
- Optimizer: Adam is used with a learning rate of 1×10^{-3} and categorical cross-entropy loss. Adam was retained for the final model because it demonstrated substantially better convergence and classification performance than SGD in the sensitivity experiment, while avoiding the need for manual learning-rate scheduling.
- Activation: ReLU is used in all hidden layers; softmax is used in the output layer to produce a normalized class-probability distribution.

Input preprocessing details: each 28×28 grayscale image is loaded directly as a 28×28 matrix, rescaled to [0, 1], and given a singleton channel dimension (channel = 1 for grayscale; channel = 3 would be

used for RGB), producing a 28×28×1 input tensor; the class labels are one-hot encoded into 10-dimensional vectors. Keras infers the batch dimension automatically at runtime.

To empirically assess the effect of the selected hyperparameters, one-factor-at-a-time sensitivity experiments were conducted while keeping the remaining settings fixed. Batch sizes of 16, 32, 50, 64, and 128 were evaluated, followed by comparisons of 3×3 and 5×5 convolutional kernels and Adam, RMSprop, and SGD optimizers. Each sensitivity experiment was conducted for five epochs using a fixed random seed of 42. The resulting validation and test performance are reported in Section 4 to provide empirical support for the selected configuration.

3. EXPERIMENTS

Experiments are conducted on the MNIST handwritten-digit dataset, which contains 60,000 training images and 10,000 testing images, each a 28×28 grayscale image of a centered, size-normalized handwritten digit (0–9). The 60,000-image training pool is further split into 55,000 images for training and 5,000 images for validation, as summarized in Table 1. For reproducibility, the 55,000/5,000 training-validation split was generated using a fixed random seed of 42; after deterministic shuffling, the same 5,000 samples were reserved for validation in the reported main experiment.

Table 1. MNIST dataset split used in this study

Subset	Sample count	Role
x_train	55,000	Training set
x_val	5,000	Validation set
x_test	10,000	Testing set

4. RESULTS AND DISCUSSION

The model was trained for 10 epochs with a batch size of 50, Adam optimizer with a learning rate of 1×10^{-3}, and categorical cross-entropy loss. All experiments were conducted using TensorFlow 2.x with Keras. Pixel values were rescaled to [0, 1] before being fed to the network. No data augmentation was applied, allowing the evaluation to focus on the effects of the proposed architecture and the investigated training hyperparameters.

4.1. Training behavior

Table 2 reports per-epoch training accuracy and loss. Validation accuracy surpasses 99% as early as epoch 2, and training accuracy crosses the same threshold by epoch 7, reaching 99.37% at epoch 10; validation accuracy peaks at 99.20% (epoch 6). This confirms the rapid-convergence behavior that motivated the architecture choice. The corresponding training-versus-validation accuracy and loss curves are shown in Figures 2 and 3, respectively. The two accuracy curves track each other closely, but the validation loss reaches its minimum of 0.0304 at epoch 4 and then rises slightly while the training loss keeps falling—a sign of mild overfitting beyond epoch 4 that the dropout layer (rate = 0.5) keeps well contained. The checkpoint with the highest validation accuracy (epoch 6) is therefore retained as the final model.

To assess the stability of the reported performance, the final configuration was additionally trained using five random seeds (42–46), while keeping the batch size at 50, kernel size at 3×3, Adam optimizer, and the 64–256–64 convolutional architecture unchanged. Across these five runs, the model achieved a mean test accuracy of 99.29% with a standard deviation of 0.06%, indicating low variation across repeated runs as shown in Figure 4. The corresponding mean test loss was 0.0284 ± 0.0026.

Table 2. Training accuracy and loss of the proposed model across 10 epochs

Epoch	Training accuracy	Training loss
1	0.9095	0.2892
2	0.9725	0.0966
3	0.9806	0.0664
4	0.9848	0.0545
5	0.9868	0.0439
6	0.9887	0.0375
7	0.9914	0.0299
8	0.9922	0.0264
9	0.9925	0.0239
10	0.9937	0.0221

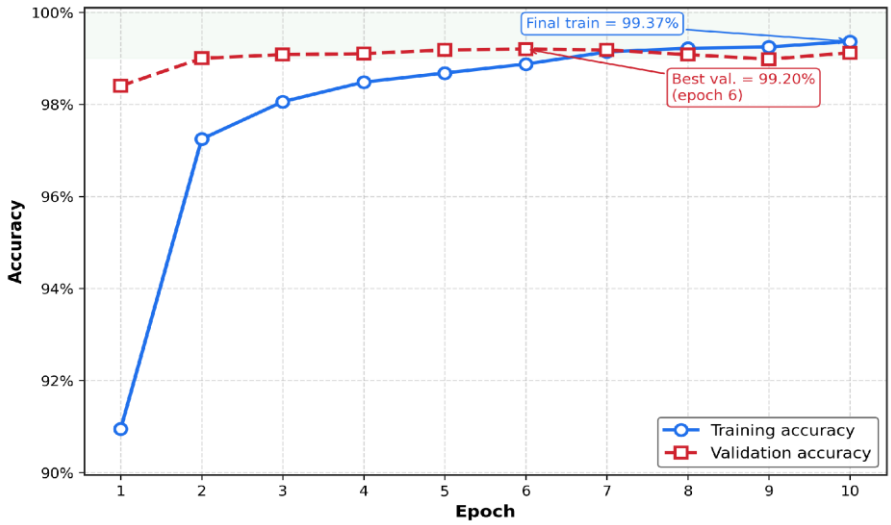

Figure 2. Training versus validation accuracy of the proposed CNN over 10 epochs (final training accuracy 99.37%; best validation accuracy 99.20% at epoch 6)

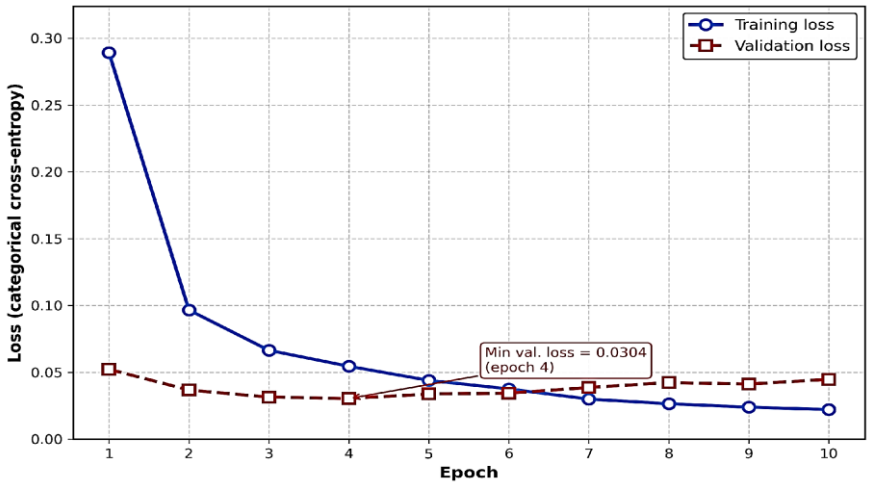

Figure 3. Training versus validation loss over the same 10 epochs. The training loss decreases monotonically; the validation loss reaches its minimum (0.0304) at epoch 4 and rises slightly thereafter, indicating the onset of mild overfitting that dropout keeps contained

4.2. Per-class metrics

Table 3 reports precision, recall and F1-score per digit class on the test set. The minimum per-class F1 is 0.985 (class 9), with class 6 the next lowest at 0.986, and the macro-averaged F1 is 0.991, confirming that performance is uniform across classes and not driven by an over-represented class.

Table 3. Per-class precision, recall and F1-score on the MNIST testing set (10,000 images)

Digit	Precision	Recall	F1-score
0	0.995	0.994	0.994
1	0.993	0.996	0.995
2	0.995	0.992	0.994
3	0.994	0.996	0.995
4	0.995	0.981	0.988
5	0.989	0.992	0.990
6	0.984	0.988	0.986
7	0.992	0.993	0.993
8	0.996	0.994	0.995
9	0.981	0.988	0.985
Macro avg.	0.991	0.991	0.991

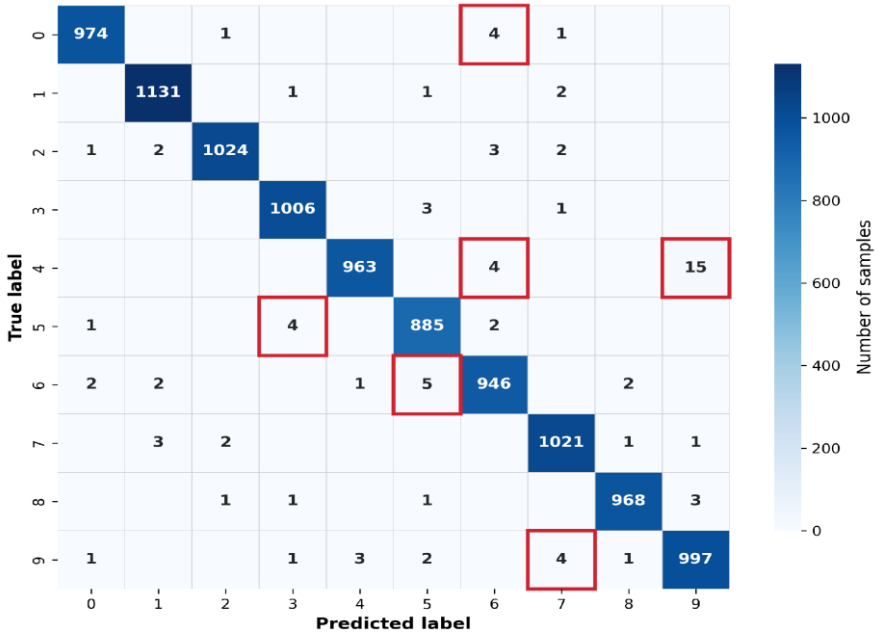

Figure 4. Confusion matrix of the proposed CNN on the MNIST test set. The matrix shows the classification distribution across all ten digit classes, with the diagonal entries representing correctly classified samples and off-diagonal entries representing misclassifications. The most noticeable errors involve visually similar digit classes, including 4–9, 6–5, and 3–5

4.3. Hyperparameter sensitivity analysis

The sensitivity analysis evaluated the effects of batch size, convolutional kernel size, and optimizer while keeping the remaining model settings fixed. The results provide empirical evidence for the behavior of the selected training configuration. As shown in Table 4, batch sizes in the range of 16–128 produced comparable validation performance, with batch size 64 achieving the lowest validation loss, while batch size 32 achieved the highest test accuracy. The 3×3 and 5×5 kernel configurations also showed comparable performance, with the 5×5 configuration obtaining slightly higher test accuracy in the five-epoch sensitivity experiment. Among the optimizers, Adam and RMSprop achieved substantially better performance than SGD under the same experimental conditions. The five-epoch setting was used only for the comparative sensitivity experiments to provide a consistent and computationally practical basis for comparing configurations, whereas the selected final configuration was subsequently trained for 10 epochs for the main performance evaluation.

Table 4. Performance comparison of different batch sizes

Experiment	Setting	Best Val.accuracy	Best Val. loss	Test accuracy	Test loss
Batch size	16	99.16%	0.031947	99.15%	0.034869
Batch size	32	99.22%	0.030433	99.35%	0.021586
Batch size	50	99.20%	0.029994	99.13%	0.029822
Batch size	64	99.18%	0.026075	99.19%	0.031930
Batch size	128	99.24%	0.027553	99.08%	0.027840
Kernel size	3×3	99.18%	0.030390	99.23%	0.025916
Kernel size	5×5	99.36%	0.024634	99.37%	0.026717
Optimizer	Adam	99.22%	0.029427	99.15%	0.029822
Optimizer	RMSprop	99.12%	0.029243	99.27%	0.030394
Optimizer	SGD	85.72%	0.587880	86.69%	0.573029

4.4. Computational complexity

In addition to predictive performance, the computational characteristics of the proposed CNN are considered to assess its suitability for resource-constrained applications as shown in Table 5. The model consists of three convolutional blocks followed by a fully connected layer and dropout. Its computational requirements are evaluated in terms of trainable parameters, model size, and inference latency. These measures provide a more objective assessment of the lightweight nature of the proposed architecture than accuracy alone.

4.5. Computational complexity

The proposed CNN contains 333,450 trainable parameters and has an FP32 model size of approximately 1.27 MB. The model requires approximately 36.62 million multiply-accumulate operations (MACs), corresponding to 73.23 million floating-point operations (FLOPs) for a single inference. In the measured execution environment, the mean inference latency was 15.69 ms, with a median of 15.33 ms and a standard deviation of 3.59 ms. These results indicate that the proposed architecture has a relatively small parameter footprint and modest computational requirements, supporting its potential for deployment in resource-constrained environments. However, the reported latency is hardware-dependent and should not be interpreted as a universal device-independent measurement.

Table 5. Computational complexity of the proposed CNN model

Metric	Value
Total parameters	333,450
Trainable parameters	333,450
Non-trainable parameters	0
Model size (FP32)	1.27 MB
MACs	36.62 million
FLOPs	73.23 million
Mean inference latency	15.69 ms
Median inference latency	15.33 ms

4.6. Comparison with the state of the art

Table 6 compares the proposed model against recently published CNN-based methods that report MNIST accuracy. Three observations stand out. First, the proposed model reaches a high training accuracy (99.37%) and a competitive test accuracy (99.15%) despite its compact 3-Conv/2-Dense design. Second, this performance is achieved within only 10 training epochs and with a comparatively small parameter footprint, while several methods in the comparison require 15–30 training epochs to reach their reported performance. Third, the CNN baselines occupy a different operating point: lower peak accuracy in exchange for significantly reduced power on neuromorphic hardware.

Table 6. Comparison of CNN-based methods for HDR on MNIST dataset

Ref.	Year	Method	Dataset	Epochs	Test accuracy (%)	Major contribution
[9]	2019	CNN	MNIST	20	98.92	Applied CNN-based feature learning for HDR on MNIST
[6]	2020	CNN for MNIST and Fashion-MNIST	MNIST	20	99.03	Explored convolutional feature extraction for digit and fashion-item classification
[8]	2022	Deep Learning Network	MNIST	20	99.08	Applied deep feature learning to improve handwritten digit classification
[1]	2023	Machine Learning with Python	MNIST	20	98.70	Presented a simple Python-based machine learning implementation for MNIST classification
[5]	2023	Deep Neural Network	MNIST	25	99.10	Applied a deep neural network approach to HDR
[7]	2023	CNN	MNIST	20	99.12	Developed a compact CNN with competitive recognition performance.
[12]	2023	Optimized Deep Learning Model	MNIST	20	99.18	Investigated CNN optimization strategies to improve MNIST classification performance
[17]	2024	Deep CNN	MNIST	20	99.50	Developed a deep convolutional architecture achieving high MNIST recognition accuracy
[18]	2024	CNN with Batch Normalization	MNIST	15	99.27	Used batch normalization to improve training convergence and stability.
[19]	2024	Sequential CNN	Handwritten Character Dataset	30	99.30	Applied a sequential CNN architecture to handwritten character recognition
[20]	2024	Comparative Neural Network Models	MNIST	20	99.22	Compared multiple neural network architectures for handwritten digit classification
Proposed	2026	Compact CNN (3 Conv + 2 Dense)	MNIST	10	99.15	Developed a compact 3-convolution/2-dense CNN and evaluated batch size, kernel size, and optimizer sensitivity

Although several existing methods achieve higher test accuracy on MNIST, the proposed model provides a competitive accuracy while using a compact architecture and achieving rapid convergence within 10 training epochs. Therefore, its primary advantage is not the highest absolute accuracy, but the balance between predictive performance, model compactness, and convergence efficiency.

5. CONCLUSION

This paper presented an compact lightweight CNN for HDR using the MNIST benchmark dataset. The proposed architecture consists of three convolutional layers followed by two fully connected layers, providing a balance between classification performance and computational efficiency. In addition to developing a compact CNN architecture, this study systematically investigated the influence of key hyperparameters, including batch size, kernel size, and optimizer, on model convergence and recognition accuracy. Experimental results demonstrated that the proposed model achieved 99.37% training accuracy and 99.15% testing accuracy after only 10 training epochs, with the highest validation accuracy of 99.20% obtained at epoch 6, indicating rapid convergence and effective generalization.

A limitation of the present study is that a dedicated architectural ablation study was not conducted to isolate the individual contribution of the third convolutional block or the 64–256–64 filter progression. The sensitivity analysis focused on batch size, kernel size, and optimizer rather than systematically removing or modifying individual convolutional blocks. Future work will therefore include controlled architectural ablations to determine the contribution of each convolutional block and filter configuration to predictive performance and computational cost.

Overall, the results indicate that careful CNN architecture design combined with empirical hyperparameter analysis can provide efficient convergence while maintaining high recognition accuracy. The measured parameter count, model size, computational cost, and inference latency further demonstrate the compact nature of the proposed model, supporting its potential applicability to HDR scenarios where computational resources and inference efficiency are important considerations. Future work will focus on evaluating the proposed architecture on more challenging handwritten character datasets, such as EMNIST, KMNIST, and Fashion-MNIST, to assess its generalization capability across diverse handwriting styles. In addition, future research will investigate lightweight attention mechanisms, transfer learning, and automated hyperparameter optimization techniques to further improve recognition performance while preserving computational efficiency. Model compression and deployment on embedded and edge devices will also be explored to facilitate real-time handwritten recognition in resource-constrained environments.

FUNDING INFORMATION

This research received no external funding. The work was conducted independently by the authors without financial support from any funding agency.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Shehla Afzal	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	
Muhammad Talha Farooqi	✓	✓	✓	✓		✓	✓	✓		✓	✓			
Irfanud Din					✓	✓	✓					✓		
Tayyaba Anees	✓				✓						✓		✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

Data Availability: The MNIST dataset used in this study is publicly available through the TensorFlow/Keras datasets API. The source code and other materials supporting the findings of this study are available in the corresponding author's GitHub repository at [https://github.com/M-Talha-Farooqi/handwritten_digit_recognition.git].

REFERENCES

[1] A. Mahesh, D. Kanagaraj, G. Thirugnanam, V. Kumar, U. Kumar, and A. Velmurugan, "Handwritten digit recognition using machine learning with python," *International Research Journal of Computer Science*, vol. 10, no. 05, pp. 168–171, 2023, doi: 10.26562/irjcs.2023.v1005.11.

[2] A. Palvanov and Y. I. Cho, "Comparisons of deep learning algorithms for MNIST in real-time environment," *INTERNATIONAL JOURNAL of FUZZY LOGIC and INTELLIGENT SYSTEMS*, vol. 18, no. 2, pp. 126–134, 2018, doi: 10.5391/ijfis.2018.18.2.126.

[3] S. Chauhan, S. Mahmood, and T. Poongodi, "Retracted: handwritten digit recognition using deep neural networks," in *2023 4th International Conference on Intelligent Engineering and Management (ICIEM)*, May 2023, pp. 1–6, doi: 10.1109/iciem59379.2023.10167391.

[4] S. S. Kadam, A. C. Adamuthe, and A. B. Patil, "CNN model for image classification on MNIST and fashion-MNIST dataset," *Journal of scientific research*, vol. 64, no. 02, pp. 374–384, 2020, doi: 10.37398/jsr.2020.640251.

[5] S. Kuma and Rahul, "Handwritten digit recognition using CNN," *International Journal of Science Technology and Engineering (IJSTE)*, vol. 11, no. 5, pp. 116–119, 2023.

[6] S. Yu, J. Liu, H. Shu, and Z. Cheng, "Handwritten digit recognition using deep learning networks," in *2022 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*, Dec. 2022, pp. 1526–1530, doi: 10.1109/tocs56154.2022.10016012.

[7] M. A. Hossain and M. M. Ali, "Recognition of handwritten digit using convolutional neural network (CNN)," *Global Journal of Computer Science and Technology*, pp. 27–33, May 2019, doi: 10.34257/gjcstdvol19is2pg27.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. https://arxiv.org/abs/1409.1556.

[9] Y. Wang *et al.*, "Improvement of MNIST image recognition based on CNN," *IOP Conference Series: Earth and Environmental Science*, vol. 428, no. 1, p. 12097, Jan. 2020, doi: 10.1088/1755-1315/428/1/012097.

[10] F. Faghihi, H. Alashwal, and A. A. Moustafa, "A synaptic pruning-based spiking neural network for hand-written digits classification," *Frontiers in Artificial Intelligence*, vol. 5, Feb. 2022, doi: 10.3389/frai.2022.680165.

[11] D. Sheng, R. Xu, Q. Wang, and C. Zhao, "Spiking neural networks for digital hand-written number recognition," in *2022 19th International SoC Design Conference (ISOCC)*, Oct. 2022, pp. 185–186, doi: 10.1109/isocc56007.2022.10031396.

[12] S. S. Ahmed, Z. Mehmood, I. A. Awan, and R. M. Yousaf, "A novel technique for handwritten digit recognition using deep learning," *Journal of Sensors*, vol. 2023, no. 1, Jan. 2023, doi: 10.1155/2023/2753941.

[13] A. Kumar, A. Aashima, A. Saini, and S. Shilpa, "MNIST-digit recognition using CNN," 2024, doi: 10.2139/ssrn.4487083.

[14] E. A. Khorsheed and A. K. Al-Sulaifanie, "Handwritten digit classification using deep learning convolutional neural network," *Journal of Soft Computing and Data Mining*, vol. 5, no. 1, 2024, doi: 10.30880/jscdm.2024.05.01.007.

[15] S. Lee and J. Rhee, "Improving test accuracy on the MNIST dataset using a simple CNN with batch normalization," *Journal of the Korea Society of Computer and Information*, vol. 29, no. 9, pp. 1–7, 2024.

[16] M. Humayun, R. Siddiqi, M. Uddin, I. A. Kandhro, M. Abdelhaq, and R. Alsaqour, "A novel methodology for offline English handwritten character recognition using ELBP-based sequential (CNN)," *Neural Computing and Applications*, Aug. 2024, doi: 10.1007/s00521-024-10206-1.

[17] F. Chen *et al.*, "Assessing four neural networks on handwritten digit recognition dataset (MNIST)," *Journal of Computer Science Research*, vol. 6, no. 3, pp. 17–22, 2024, doi: 10.30564/jcsr.v6i3.6804.

[18] K. Ladrham and H. Gueddah, "Advanced OCR for digits exploring CNN for optimal performance," *Procedia Computer Science*, vol. 251, pp. 734–739, 2024, doi: 10.1016/j.procs.2024.11.177.

[19] X. Zhang, "Handwritten digit recognition based on deep learning techniques," *Applied and Computational Engineering*, vol. 36, no. 1, pp. 7–14, Feb. 2024, doi: 10.54254/2755-2721/36/20230416.

[20] M. A. Khan, F. Ahmad, K. Khan, and M. Khan, "Pashto language handwritten numeral classification using convolutional neural networks," in *Forthcoming Networks and Sustainability in the AIoT Era*, Springer Nature Switzerland, 2024, pp. 287–297.

[21] D. Ben Noureddine, "Handwritten digit recognition: comparative analysis of Ml, CNN, vision transformer, and hybrid models on the mnist dataset," 2024, doi: 10.2139/ssrn.4970809.

[22] S. K, M. M, N. S R, and N. Krishna R, "Handwritten digit and Roman string recognition using gated CNN with federated learning," *International Journal of Hybrid Intelligent Systems*, vol. 21, no. 2, pp. 163–175, 2024, doi: 10.3233/his-240030.

[23] Y. Kong, "Research and application of constructing image recognition models for archival handwriting," *Journal of Computational Methods in Sciences and Engineering*, vol. 25, no. 2, pp. 1667–1677, Nov. 2024, doi: 10.1177/14727978241302442.

[24] K. Banaszewska and M. Plechawska-Wójcik, "Comparative analysis of CNN models for handwritten digit recognition," *Journal of Computer Sciences Institute*, vol. 32, pp. 179–185, 2024, doi: 10.35784/jcsi.6239.

[25] J. Jiang, "Research on the application of convolutional neural networks on MNIST datasets," *Applied and Computational Engineering*, vol. 109, no. 1, pp. 189–196, Dec. 2024, doi: 10.54254/2755-2721/2024.18132.

[26] Y. Li, "Handwritten digit recognition based on TensorFlow framework," *Applied and Computational Engineering*, vol. 157, no. 1, pp. 172–178, 2025, doi: 10.54254/2755-2721/2025.po24694.

BIOGRAPHIES OF AUTHORS



Shehla Afzal is a Lecturer in the Department of software Engineering at University of Management and Technology, Pakistan. She received her M.S. degree in Computer Science from UAF, Pakistan and is currently pursuing a Ph.D. in Computer Science. Her research focuses on artificial intelligence, deep learning, machine learning, computer vision, handwritten digit recognition, renewable energy forecasting, and the Internet of Things (IoT). She has authored and co-authored research papers in the fields of deep learning and intelligent systems. Her current research aims to develop efficient AI-based models for handwritten digit recognition and intelligent solar energy management using advanced deep learning techniques. She can be contacted at the email: shehla.afzal@umt.edu.pk.



Muhammad Talha Farooqi received the B.S. degree in Software Engineering from the University of Management and Technology (UMT), Lahore, Pakistan. His research interests include deep learning, computer vision, and LLMs. He has worked on AI-based surveillance and real-world automation projects, with a focus on public safety and intelligent monitoring applications. He is currently working as an AI Engineer at a private firm in Lahore. He can be contacted at the email: the.mtalhafarooqi@gmail.com.



Dr. Irfanud Din is a Professor and Head of the Department of Computer Science at Superior University Lahore, Pakistan, where he has been serving since 2018. He also served as the Director of the Office of Research, Innovation and Commercialization (ORIC) from 2021 to 2022. He received his Ph.D. degree in Wireless Communication from Incheon National University, South Korea, and completed his postdoctoral research at the Korea Advanced Institute of Science and Technology (KAIST), South Korea. His previous academic appointments include Associate Professor at INHA University in Tashkent, Uzbekistan, Visiting Lecturer at Incheon National University, South Korea. His research interests include wireless communications, visible light communication (VLC), LTE/Wi-Fi coexistence, embedded systems, computer vision, and intelligent networking. He has authored numerous research articles in high-impact international journals, including IEEE photonics technology letters, IEEE access, lighting research and technology, and computers, materials and continua. He can be contacted at email: irfan@newuu.uz.



Dr. Tayyaba Anees joined Vienna University of Technology, Vienna, Austria for her PhD in 2008. She completed her PhD in 2012. Her PhD dissertation is in the area of Service-oriented architecture and web services availability domain. She has worked as a project assistant in Vienna University of Technology, Vienna, Austria. She has worked at Punjab University College of Information Technology, University of the Punjab, Lahore, Pakistan for over 6 years. She has taught courses including distributed and parallel systems, research trends in software engineering, software quality assurance, software project management, introduction to computing, programming fundamentals, object oriented programming, software engineering, internet programming and web technology to PhD, MS, Msc and BS students. She is currently working as Associate Professor/Chairperson Department of Software Engineering, University of Management and Technology, Lahore, Pakistan. She can be contacted at email: se.cod@umt.edu.pk.