And., Intl.,J., Appl Sci., Eng., Tech, Vol 06, No 02, July 2026
e-ISSN: 2797-0442

A Comparative Study of Deep Learning Models for Image Classification: Simple MLP, Deep MLP, Basic CNN, LeNet CNN

Baffoe Nicholas*, Jephthah Kwame Lanor
Department of Computer and Electrical Engineering, University of Energy and Natural Resources, Sunyani, Ghana
*Corresponding author, email: baffoenicholas1@gmail.com

Abstract—This study presents a comparative analysis of four deep learning architectures: Simple Multi-Layer Perceptron (MLP), Deep MLP, Basic Convolutional Neural Network (CNN), and LeNet Convolutional Neural Network (CNN). The models are evaluated on the MNIST handwritten digit dataset under identical experimental conditions using CPU-only hardware. Each model is assessed based on classification accuracy, training time, and number of trainable parameters. Experimental results demonstrate that convolutional architectures consistently outperform fully connected architectures in classification accuracy, with the Basic CNN achieving the highest validation accuracy of 99.15% and the LeNet CNN offering the best balance between performance and efficiency with only 61,706 trainable parameters. The findings confirm that architectural design has a greater influence on model performance than the number of layers or parameters alone. This study provides a comprehensive review of the relevant literature, detailed architectural descriptions, experimental methodology, and a thorough analysis of results to support the comparative conclusions.

Keywords: Deep Learning, Convolutional Neural Network, Multi-layer Perceptron, MNIST, PyTorch, Model Comparison.

This article *is licensed under the CC–BY-SA license.*

1. Introduction

Deep learning has emerged as one of the most transformative areas of artificial intelligence, enabling machines to learn complex representations directly from raw data without the need for manual feature engineering. Since the early successes of deep learning models on large-scale image recognition tasks in the early 2010s, the field has experienced rapid growth in both research activity and practical application [1]. This progress has been driven by improvements in training algorithms such as backpropagation, increases in computational power, and the availability of large benchmark datasets.

Neural networks, which form the foundation of deep learning, have evolved considerably over the past few decades. Two major families of neural network architectures that have been widely studied and deployed are the Multi-Layer Perceptron (MLP) and the Convolutional Neural Network (CNN) [2]. MLPs are among the earliest neural network architectures and typically consist of an input layer, one or more hidden layers, and an output layer, with neurons fully connected across adjacent layers. Although MLPs are effective for general-purpose function approximation, they do not naturally exploit the spatial structure present in image data. CNNs, introduced by LeCun et al. [3] and popularized through the LeNet architecture, address this limitation by using convolutional operations to learn local spatial features through shared-weight filters. This characteristic makes CNNs especially suitable for image classification and other computer vision tasks.

Despite the well-established theoretical advantages of CNNs over MLPs for image processing tasks, selecting the most appropriate architecture for a given problem remains a non-trivial challenge. Different architectures vary in performance depending on their structural design, depth, number of parameters, and the nature of the input data. In image classification, a model may achieve high accuracy

Received: April 23, 2026, Revised: May 12, 2026, Accepted: June 26, 2026
https://doi.org/10.25077/aijaset.v6i2.343

at the cost of increased computational complexity, while another may be more computationally efficient but less accurate. This trade-off is particularly important in resource-constrained environments, such as systems operating on CPU hardware without GPU acceleration [5], where recent studies have focused on lightweight and computationally efficient deep learning architectures for edge and low-power systems [17],[18],[19]. Recent advances in lightweight convolutional neural networks and efficient model design have demonstrated that high classification accuracy can be achieved while reducing computational cost and memory usage [20], [21]. Systematic empirical comparisons conducted under identical experimental conditions therefore remain valuable because they provide clearer evidence of the effects of architectural design, independent of differences in training procedures, datasets, or hardware [4].

This study addresses this gap by implementing, training, and evaluating four widely recognized deep learning models: Simple MLP, Deep MLP, Basic CNN, and LeNet CNN. The study aims to provide a comparative analysis of these four architectures for image classification under identical experimental conditions using the MNIST dataset. All four models are implemented in the PyTorch framework and trained using the same configuration, including the optimizer, loss function, learning rate, batch size, and number of epochs. Their performance is evaluated using three measurable metrics: classification accuracy, training time, and number of trainable parameters. These metrics provide a clear basis for comparing predictive performance with computational efficiency and for identifying the most suitable model within the scope of this study.

The work is limited to handwritten digit classification on the MNIST benchmark dataset [7] and does not include more advanced architectures such as ResNet, VGGNet, or Transformer-based models. Other performance indicators such as precision, recall, and F1-score, while potentially informative, fall outside the scope of this study. Because the MNIST dataset contains approximately balanced samples across digit classes, classification accuracy is considered a sufficient and reliable metric for this evaluation. GPU acceleration is also excluded, as the experimental environment is constrained to CPU-only computation [5]. Within these boundaries, the study offers a focused comparison of fully connected and convolutional architectures and highlights the practical trade-offs involved in model selection.

This study contributes to the understanding of deep learning model selection and evaluation in several ways. First, it provides an empirical basis for comparing fully connected and convolutional architectures under controlled and reproducible experimental conditions. In addition, it demonstrates the practical trade-offs between model accuracy, computational cost, and architectural efficiency, which is particularly relevant for practitioners working in resource-constrained environments. Also, the use of the MNIST benchmark dataset ensures that the results are directly comparable to a broad body of existing literature [6], thereby supporting further research and replication. Finally, the detailed documentation of the experimental setup, training procedure, and analysis methodology provides a reusable framework that can be extended to other datasets and architectures in future work.

2. Methods

2.1 Experimental Setup

All experiments were conducted on an HP EliteBook x360 830 G6 personal computer equipped with an Intel Core i5-8265U processor (1.60 GHz, 8 logical CPUs) and 16 GB of RAM, running Windows 11 Pro 64-bit. No GPU acceleration was available, and all model training and evaluation were therefore performed in CPU-only mode. CPU-only evaluation remains relevant for studying efficient deep learning deployment in constrained computing environments and edge-based systems [19], [20]. The software environment was based on Python 3.11, with PyTorch [15] used as the primary deep learning framework. Torchvision 0.25.0 was used for dataset handling, NumPy 2.4.3 for numerical computation, and Pillow 12.1.1 for image processing.

2.1 Dataset

The MNIST handwritten digit dataset [6] was used for all experiments. It contains 70,000 grayscale images of handwritten digits belonging to 10 classes (0-9), with each image having a resolution of 28 × 28 pixels. The dataset is divided into 60,000 training images and 10,000 test images. Before training, the images were transformed into floating-point tensors in the range [0.0, 1.0] using ToTensor(). The data were then normalized [16] using the mean (0.1307) and standard deviation (0.3081) of the MNIST training set. This preprocessing step helps stabilize the input distribution and supports efficient model training [6].

2.3 Model Architectures

Four deep learning models were implemented for comparison. The Simple MLP, Deep MLP and Basic CNN architectures used in this study were custom baseline configurations designed specifically for comparative evaluation under identical experimental conditions, while the LeNet CNN follows the standard LeNet-5 architecture proposed by LeCun et al. [3].

The **Simple MLP** consists of a flattened 784-unit input layer, one hidden layer with 256 units, a ReLU activation function [10],[9] and a 10-unit output layer, with approximately 203,530 trainable parameters.

The **Deep MLP** extends this structure by using four hidden layers with 512, 256, 128, and 64 units respectively. ReLU activations [10] are applied after each hidden layer, and Dropout with a rate of 0.3 is included for regularization [21]. This model contains approximately 575,050 trainable parameters.

The **Basic CNN** consists of two convolutional layers with 16 and 32 filters of size 3 × 3, each followed by ReLU activation and 2 × 2 max-pooling [10]. The extracted features are then passed through a fully connected layer with 128 units before the final 10-class output layer. This model contains approximately 206,922 trainable parameters. The use of convolution and pooling is consistent with established CNN design principles for vision tasks [21].

The **LeNet CNN** follows the LeNet-5 architecture proposed by LeCun et al. [3]. It consists of two convolutional layers with 6 and 16 filters of size 5 × 5, average-pooling layers, and three fully connected layers with 120, 84, and 10 units. Tanh activation functions are used throughout the network, and the model contains approximately 61,706 trainable parameters [13].

Table 1. A summary of the four architectures

Model	Input	Hidden Layers	Activation	Parameters
Simple MLP	784	256	ReLU	203,530
Deep MLP	784	512-256-128-64	ReLU+Dropout	575,050
Basic CNN	28×28×1	Conv(16,32) +FC(128)	ReLU+MaxPool	206,922
Lenet CNN	28×28×1	Conv(6,16) +FC(120,84)	Tanh+AvgPool	61,706

2.4 Training Configuration

To ensure a fair comparison, all four models were trained under identical conditions. Each model was trained for 10 epochs using the Adam optimizer [7] with a learning rate of 0.001. Cross-Entropy Loss was used as the objective function for the multi-class classification task. A batch size of 64 was selected to balance computational efficiency with stable gradient estimation. Data shuffling was enabled for the training set only.

2.5 Evaluation Metrics

Three metrics were used to evaluate model performance. **Classification accuracy** was used as the primary performance metric and was defined as the percentage of correctly classified samples in the test set. **Training time** was measured as the total wall-clock duration required to complete training for each model and was used as an indicator of computational efficiency. **Number of trainable parameters** was used to represent model complexity and memory requirement. This value was computed as the total number of parameters requiring gradient updates during training.

2.6 Experimental Procedure

The MNIST dataset was first downloaded and preprocessed as described in Section 2.2. Each model was then implemented in PyTorch [15] as a subclass of torch.nn.Module, with the layer definitions specified in the initialization stage and the forward computation defined in the forward method.

Each model was instantiated on the CPU device, and its trainable parameter count was recorded before training. The models were then trained independently for 10 epochs. At the end of each epoch, training accuracy, validation accuracy, training loss, validation loss, and epoch duration were recorded. After training, the best validation accuracy, total training time, and parameter count for each model were extracted and stored for comparative analysis.

Several experimental constraints should be noted. First, all training was performed in a CPU-only environment, which increased computation time, especially for the convolutional models. Second, training was limited to 10 epochs, which may not have been sufficient for every model to reach full convergence. Third, the evaluation was conducted on only one dataset, so the generalizability of the findings to other image classification tasks was not assessed. Finally, no hyperparameter tuning was performed, meaning each model was evaluated under the same fixed training configuration.

3. Result and Discussion

The four deep learning models, namely Simple MLP, Deep MLP, Basic CNN, and LeNet CNN, were trained for 10 epochs on the MNIST dataset under identical experimental conditions. Their final performance is summarized in Table 2, which reports the best validation accuracy, total training time, number of trainable parameters and efficiency ratio for each model. The efficiency ratio was computed as validation accuracy divided by the number of trainable parameters and scaled by 10^{-3} for readability.

Table 2. Final performance summary of all four models after 10 training epochs

Model	Parameters	Best val. accuracy (%)	Efficiency Ratio (acc/param $\times 10^{-3}$)	Total training time (min)
Simple MLP	203,530	98.01	0.482	7.6
Deep MLP	575,050	98.10	0.171	6.6
Basic CNN	206,922	99.15	0.479	11.3
Lenet CNN	61,706	99.04	1.606	7.9

Table 2 shows that the Basic CNN achieved the highest best validation accuracy of 99.15%, followed by the LeNet CNN at 99.04%, the Deep MLP at 98.10%, and the Simple MLP at 98.01%. In

terms of computational cost, the Basic CNN also required the longest total training time at 11.3 minutes, whereas the Deep MLP recorded the shortest training time at 6.6 minutes. The LeNet CNN had the smallest number of trainable parameters at 61,706, while the Deep MLP had the largest at 575,050. These results indicate that the most accurate model was not the most efficient in training time, and that the largest model was not necessarily the best-performing one.

Table 3. Epoch-by-epoch validation accuracy (%) for all four models on the MNIST test set

Epoch	Simple MLP	Deep MLP	Basic CNN	LeNet CNN
1	96.49	95.91	97.63	97.70
2	96.94	96.58	98.62	98.30
3	97.74	97.33	98.92	98.49
4	97.25	97.46	99.05	98.66
5	97.92	97.62	98.94	98.70
6	97.96	97.96	99.10	98.76
7	97.72	97.86	99.02	98.73
8	98.01	98.05	99.01	98.99
9	97.81	98.10	99.09	99.04
10	98.00	98.03	99.15	98.87

The epoch-by-epoch validation accuracy values in Table 3 show the convergence behaviour of the four models over the training period. Both convolutional models reached high validation accuracy early in training. The Basic CNN exceeded 98.60% by the second epoch, while the LeNet CNN reached 98.30% at the same stage. By contrast, the MLP-based models improved more gradually and showed greater fluctuation across epochs. All four models had largely stabilised by about the eighth epoch, with only small changes afterwards. This suggests that 10 epochs were sufficient to reveal the main performance characteristics of each architecture under the selected configuration.

The validation accuracy trends across the 10 training epochs are shown in Figure 1. The figure shows that the two CNN-based models maintained a clear advantage over the MLP-based models throughout training. The Basic CNN remained the strongest performer across most epochs, while the LeNet CNN followed closely behind. The two MLP models converged at lower validation accuracy levels and exhibited more variation before stabilising. This visual trend is consistent with the final performance values reported in Table 3 and supports the conclusion that convolutional architectures are better suited to this image classification task.

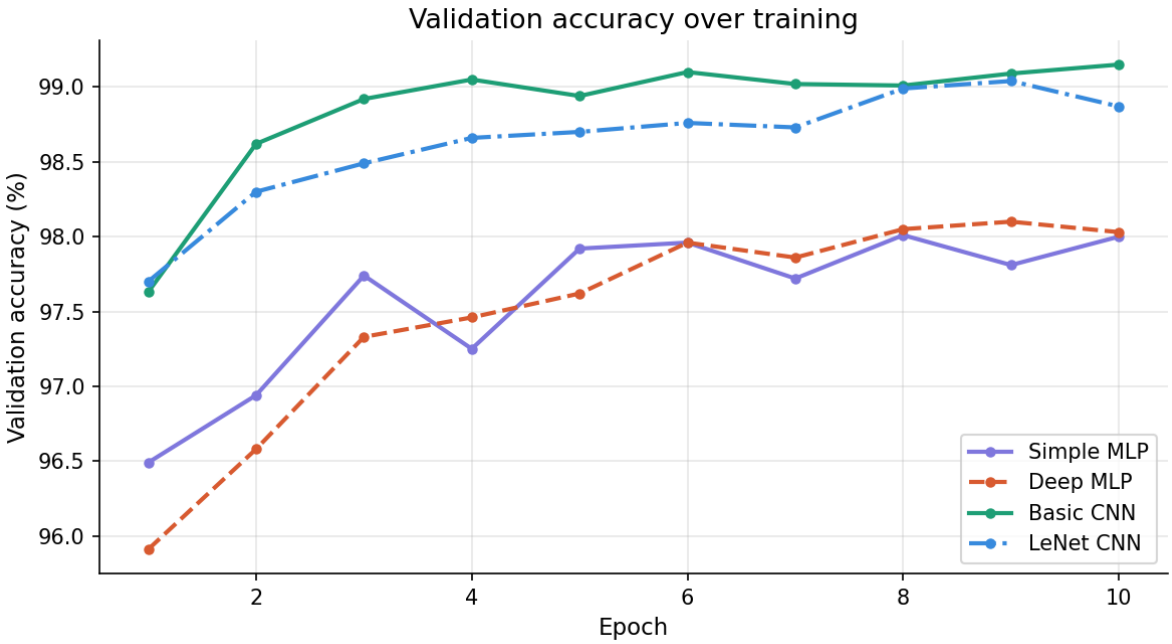


Figure 1. Validation accuracy of the four models across 10 training epochs on the MNIST test set.

The results demonstrate a clear and consistent accuracy advantage for the convolutional architectures over the fully connected architectures. Although the absolute difference between the CNN models and the MLP models is approximately one percentage point, this margin is meaningful on the MNIST benchmark, where performance differences of this scale between model families have been consistently reported in the literature [14]. The main reason for this difference lies in the way the two architecture families process image data. MLPs operate on flattened one-dimensional vectors and therefore do not preserve spatial relationships between neighboring pixels. CNNs, in contrast, apply convolutional filters across the image and are able to learn local features such as edges, curves, and stroke patterns that are directly relevant to handwritten digit recognition [3]. This spatial sensitivity explains the consistent performance advantage of the CNN models across all training epochs.

Within the MLP family, increasing the network depth produced only a marginal improvement. The Deep MLP outperformed the Simple MLP by just 0.09 percentage points despite having far more parameters. This finding supports observations in the literature that deeper fully connected networks offer limited benefit for image classification tasks when the model still lacks a mechanism for exploiting spatial structure [20]. The use of Dropout in the Deep MLP was important for reducing overfitting and improving generalization [21], but it did not overcome the structural limitation of the architecture. Within the CNN family, the Basic CNN slightly outperformed the LeNet CNN. This is consistent with prior findings that ReLU activations often support faster convergence and strong generalization [19], and that smaller 3 × 3 convolutional filters can provide an efficient parameterization for learning local image features compared with the 5 × 5 filters used in the original LeNet design [12].

Training time revealed an important practical trade-off between accuracy and computational cost. The total training time required by each model is shown in Figure 2. The Basic CNN required the longest training time at 11.3 minutes, whereas the Deep MLP required the shortest at 6.6 minutes. The LeNet CNN, despite being convolutional, required only 7.9 minutes, which was much closer to the training times of the MLP models than to that of the Basic CNN.

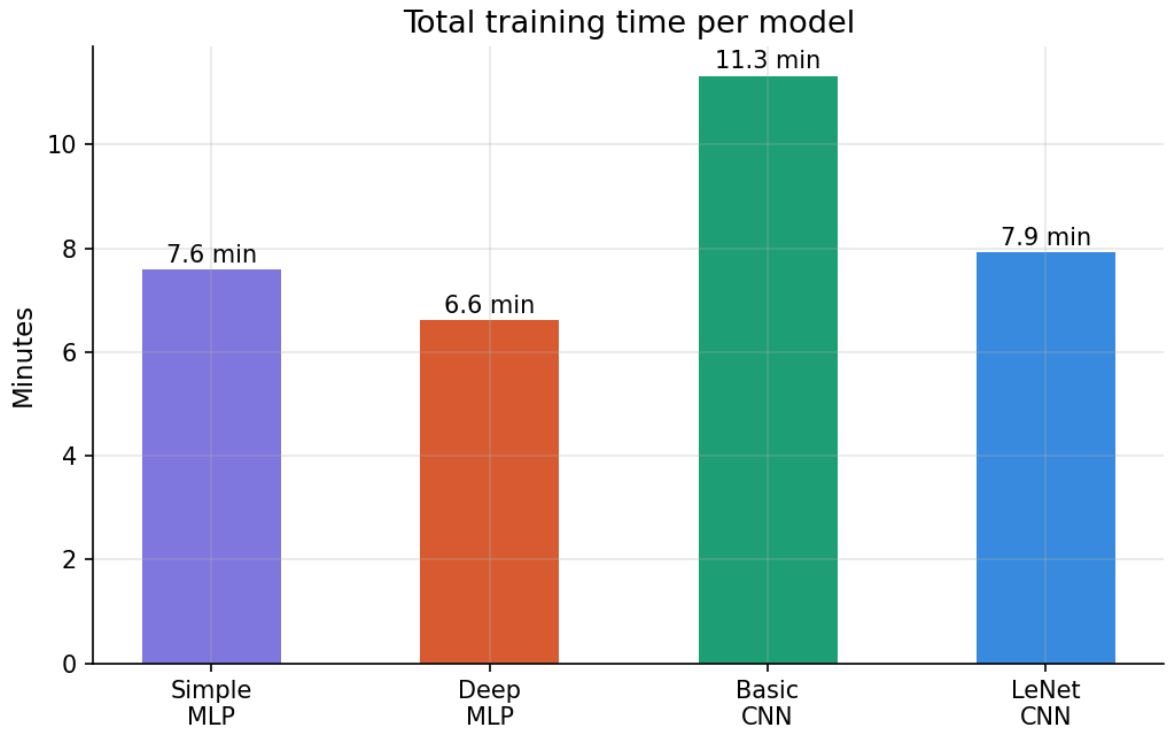


Figure 2. Total training time of the four models under CPU-only execution.

The longer training time of the Basic CNN is consistent with the higher computational cost of convolutional operations, particularly in a CPU-only environment where such operations are less efficiently parallelized than on GPU hardware [5]. The LeNet CNN trained faster than the Basic CNN because it used substantially fewer parameters and smaller feature maps. An interesting result is that the Deep MLP trained faster than the Simple MLP despite having nearly three times as many parameters. This behavior may be influenced by implementation-level optimization differences, matrix operation efficiency, CPU scheduling variability, and the stochastic effects of dropout during training. Overall, these results show that convolutional models provide better classification performance, but they may also introduce additional computational overhead under limited hardware conditions. These findings are consistent with recent studies on lightweight CNN architectures, which report that computational efficiency and parameter optimization are critical factors in practical deep learning deployment, particularly on CPU-based and edge-computing systems [17], [18], [19].

The relationship between parameter counts and accuracy provides another important insight. The Deep MLP had the largest number of trainable parameters at 575,050, yet it achieved only 98.10% accuracy. In contrast, the LeNet CNN used only 61,706 parameters, which is approximately 9.3 times fewer than the Deep MLP, but still achieved 99.04% accuracy. This shows that a larger parameter count does not necessarily result in better performance. Table 2 quantifies this relationship through the accuracy-to-parameter ratio for each model.

The efficiency values in Table 2 show that the LeNet CNN achieved the most favorable accuracy-per-parameter trade-off, with a ratio of 1.606×10^{-3}, which is more than three times that of the next-best model. The Deep MLP had the lowest efficiency ratio at 0.171×10^{-3}, indicating that its large number of parameters was not being used efficiently for this task. These results support the conclusion that architectural suitability is more important than raw model size for image classification. They are consistent with the findings of Rawat and Wang [14], who observed that architecture selection should be guided by the computational requirements of the target environment, and with the earlier work of LeCun et al. [3], which demonstrated the value of convolutional inductive bias for image-based tasks.

Several limitations should be considered when interpreting these findings. First, all training was performed on CPU hardware, and the reported times may therefore include variability caused by operating system scheduling and background processes. Second, the study was limited to a single benchmark dataset, and the relative performance of the models may differ on more complex datasets such as CIFAR-10 or ImageNet [8],[11], [12]. Third, all models were trained for only 10 epochs, so some architectures may not have reached their full performance potential. Finally, no hyperparameter tuning was performed, and some models may yield different results under settings tailored to their specific design. Recent lightweight CNN studies have also shown that architectural performance may vary substantially across datasets with higher visual complexity and larger image dimensions [18], [21].

4. Conclusion

This study presented a comparative analysis of four deep learning architectures, namely Simple Multi-Layer Perceptron (MLP), Deep MLP, Basic Convolutional Neural Network (CNN), and LeNet CNN, for handwritten digit classification using the MNIST dataset. All models were implemented in PyTorch and evaluated under identical experimental conditions on CPU hardware using classification accuracy, training time, and number of trainable parameters as evaluation metrics.

The results showed that all models achieved high classification performance, with accuracies exceeding 98%. However, consistent differences were observed between the architectural types. The CNN-based models outperformed the MLP-based models, with the Basic CNN achieving the highest validation accuracy of 99.15%, followed closely by the LeNet CNN at 99.04%. This confirms the effectiveness of convolutional architectures for image classification tasks, as they are able to capture spatial features such as edges, curves, and stroke patterns that are not explicitly preserved in fully connected networks [3].

The LeNet CNN demonstrated the highest parameter efficiency, achieving near state-of-the-art accuracy with significantly fewer parameters than the other models. In contrast, the Deep MLP, despite having the largest number of parameters, produced only a marginal improvement over the Simple MLP. This indicates that increasing model depth and parameter count alone does not necessarily lead to improved performance when the architecture is not well suited to the structure of the data. In all, the findings confirm that architectural design is a more important factor than model size for image classification tasks. Convolutional neural networks provide a clear advantage over fully connected networks due to their ability to exploit the spatial structure of image data. The findings highlight the importance of selecting architectures that align with the structural properties of the data and the computational constraints of the deployment environment [3].

Future work may extend this analysis to more complex datasets and modern architectures, as well as explore the impact of hyperparameter optimization and GPU-accelerated training on model performance.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[3] 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, Nov. 1998.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[5] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, “Efficient processing of deep neural networks: A

tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[6] Y. LeCun and C. Cortes, "MNIST handwritten digit database,"2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, vol. 25, pp. 1097–1105.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[10] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines,"in *Proc. 27th Int. Conf. Machine Learning (ICML)*, Haifa, Israel, 2010, pp. 807–814.

[11] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[12] 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.

[13] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision, " in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, Paris, France, 2010, pp. 253–256.

[14] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, Sep. 2017.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems, vol. 32, 2019.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift, ," in *Proc. 32nd Int. Conf. Machine Learning (ICML)*, Lille, France, 2015, pp. 448–456.

[17] C. Cui, T. Gao, S. Wei, Y. Du, R. Guo, S. Dong, B. Lu, Y. Zhou, X. Lv, Q. Liu, X. Hu, D. Yu, and Y. Ma, "PP-LCNet: A lightweight CPU convolutional neural network," arXiv preprint arXiv:2109.15099, 2021.

[18] Y. Liu, J. Xue, D. Li, W. Zhang, T. K. Chiew, and Z. Xu, "Image recognition based on lightweight convolutional neural network: Recent advances," *Image Vis. Comput.*, vol. 146, p. 105037, Jun. 2024.

[19] R.-Y. Ju, T.-Y. Lin, J.-H. Jian, and J.-S. Chiang, "Efficient convolutional neural networks on Raspberry Pi for image classification," *arXiv preprint arXiv*:2204.00943, 2022.

[20] S. M. Raza, S. M. H. Abidi, M. Masuduzzaman, and S. Y. Shin, "Lightweight deep learning for visual perception: A survey of models, compression strategies, and edge deployment challenges,"

Neurocomputing, vol. 656, p. 131357, 2025.

[21] O. E. Korkmaz, “Revisiting convolutional design for efficient CNN architectures in edge-aware applications,” Scientific Reports, vol. 15, Art. no. 44351, Nov. 2025.