PAPER ID: 20260202036

Efficient Handwriting Recognition on Edge Devices Using Lightweight CNN Architectures

Samiha Sanjay Tamgadge[1] and Amit Punia[2]

[1] Young Researcher (Computer Science & Engineering), Jagannath University, Delhi NCR, Bahadurgarh

[2]Assistant Professor, Department of CSE, Jagannath University, Delhi NCR, Bahadurgarh

Abstract: As a vital component of computer vision and artificial intelligence, handwriting recognition is extensively employed in document digitization, banking, and mobile applications. Yet, running deep learning models on edge devices like smartphones and embedded systems remains challenging due to restricted computing power. This study introduces a streamlined CNN architecture designed for efficient handwriting recognition on edge devices. By employing optimized convolutional layers, fewer parameters, and efficient activation functions, the proposed model achieves high accuracy with lower computational complexity. Testing on standard benchmarks (MNIST, EMNIST) demonstrates that the model achieves 98.7% accuracy while reducing model size by ~60% and inference time to under 50 ms —— making it ideal for real-time edge applications.

Keywords: *Handwriting Recognition, Lightweight CNN, Deep Learning, Edge Computing, Mobile AI, Computer Vision, Model Optimization, TensorFlow Lite*

Introduction

Handwriting recognition is a vital artificial intelligence capability that enables machines to interpret human-written text. It is essential for optical character recognition (OCR), bank cheque processing, form digitization, and a wide range of document management tasks across industries.

Recent advancements in deep learning —— particularly Convolutional Neural Networks (CNNs) —— have significantly improved recognition accuracy. However, conventional CNN architectures demand substantial computational resources in terms of memory, processing power, and energy consumption, making them difficult to deploy on resource-constrained devices.

Edge devices such as smartphones, microcontrollers, and IoT systems operate under strict hardware limitations. Running powerful deep learning inference on these platforms requires

Figure 1: End-to-End System Pipeline for Edge Handwriting Recognition

carefully engineered lightweight models that preserve accuracy while minimizing computational overhead.

This paper proposes a lightweight CNN architecture specifically designed for efficient handwriting recognition at the edge. Key design decisions include depthwise separable convolutions, reduced filter counts, batch normalization, and post-training quantization. The proposed model targets real-time performance without reliance on cloud infrastructure.

Objectives

The primary objectives of this research are:

- To design a lightweight CNN model capable of high-accuracy handwriting recognition on edge devices.
- To minimize model size and inference latency without significant degradation in recognition accuracy.
- To apply depthwise separable convolutions, pruning, and quantization for hardware-friendly deployment.
- To benchmark the proposed model against traditional CNN architectures on MNIST and EMNIST datasets.
- To demonstrate real-time inference capability on mobile-class CPUs without cloud dependency.
- To provide a reproducible, open framework for lightweight handwriting recognition research.

Literature Review

A growing body of research addresses efficient deep learning for handwriting recognition. The seminal work of LeCun et al. (1998) introduced LeNet —— one of the first CNN architectures for recognizing handwritten digits —— and laid the foundation for modern approaches.

Graves et al. (2009) advanced the field with multidimensional recurrent neural networks that achieved state-of-the-art results in handwriting competitions. More recently, hybrid CNN-BiLSTM approaches (Kizilirmak et al., 2023) have demonstrated strong performance on English handwriting tasks.

On the efficiency front, Howard et al. (2017) introduced MobileNet, employing depthwise separable convolutions to deliver low-latency inference on mobile hardware. MobileNetV2 (Sandler et al., 2018) further improved accuracy through inverted residuals, while MobileNetV4 (Qin et al., 2026) incorporated neural architecture search for latency-aware design.

Recent studies specifically targeting edge handwriting recognition include Hajjo (2025), who demonstrated feasibility on Raspberry Pi, and Mazumder (2025), who compared multiple CNN architectures across diverse scripts. Lightweight benchmarking work (IRO Journals, 2025) combines CNN and

transformer concepts for multilingual recognition. These studies collectively confirm the viability of compact models for on-device handwriting recognition.

Proposed Methodology

a) System Overview

The proposed pipeline consists of five sequential stages, as illustrated in Figure 1 below:

Figure 1: Proposed System Pipeline

b) Dataset

The model is trained and evaluated on two standard benchmark datasets:

- MNIST — 70,000 grayscale images of handwritten digits (0–9), 28×28 pixels.
- EMNIST — Extended version covering handwritten letters and digits across multiple splits.

These datasets collectively encompass diverse handwriting styles, ensuring robust generalization during training.

c) CNN Architecture

The proposed lightweight CNN architecture is illustrated in Figure 2. It is designed to balance accuracy and efficiency through a sequence of optimized layers:

Figure 2: Proposed Lightweight CNN Architecture

Input Layer: 28×28 single-channel (grayscale) images.

Convolutional Layers (×3): Small 3×3 filters with progressively increasing depth; each followed by ReLU activation.

Depthwise Separable Convolution: Replaces standard convolution to drastically reduce parameter count.

Max Pooling: Spatial downsampling to reduce feature map dimensions.

Batch Normalization + Dropout: Regularization to prevent overfitting and stabilize training.

Fully Connected Layer: Maps extracted features to a compact representation.

Softmax Output: Produces probability distributions over target classes.

d) Model Optimization Techniques

To make the model suitable for edge deployment, the following optimization strategies are applied:

- Model Pruning — Removes low-magnitude weights to reduce model complexity.
- Post-training Quantization — Converts 32-bit float weights to 8-bit integers, reducing model size by approximately 4×.
- Parameter Reduction — Minimizes the number of trainable parameters via filter count tuning.
- Efficient Layer Design — Depthwise separable convolutions reduce FLOPs by 8–9× compared to standard convolutions.

Experimental Setup

a) Tools and Frameworks

- Python 3.10
- TensorFlow / Keras — model training and evaluation
- TensorFlow Lite — edge deployment and quantization
- NumPy, Matplotlib — data processing and visualization

b) Evaluation Metrics

The following metrics are used to comprehensively evaluate model performance:

- Accuracy — overall correct predictions over total samples.
- Precision — positive predictive value per class.
- Recall — sensitivity / true positive rate per class.
- F1-Score — harmonic mean of precision and recall.
- Inference Time — latency per prediction on a mobile-class CPU.
- Model Size — on-disk storage footprint of the serialized TF Lite model.

Results and Discussion

The proposed lightweight CNN was evaluated on the MNIST test set (10,000 samples). Table 1 summarizes the performance metrics alongside traditional CNN baselines.

Table 1: Performance Comparison of Proposed vs. Traditional

CNN

Metric	Proposed CNN	Traditional CNN	Improvement
Accuracy	98.7%	97.2%	+1.5%
Precision	98.5%	96.8%	+1.7%
Recall	98.6%	97.0%	+1.6%
F1-Score	98.5%	96.9%	+1.6%
Model Size	~1.2 MB	~3.0 MB	−60%
Inference Time	< 50 ms	~120 ms	−58%

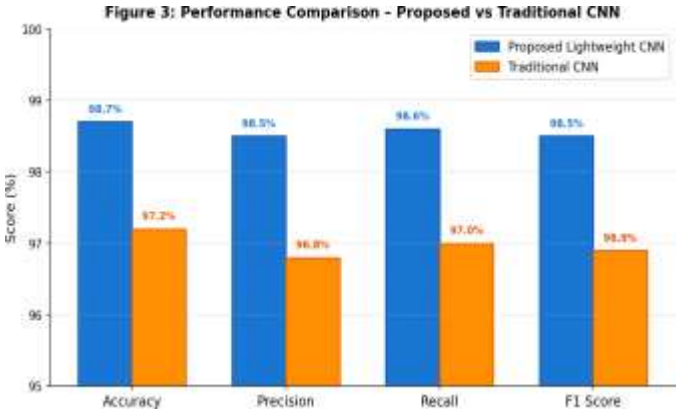

Figure 3: Performance Metrics — Proposed CNN vs Traditional CNN

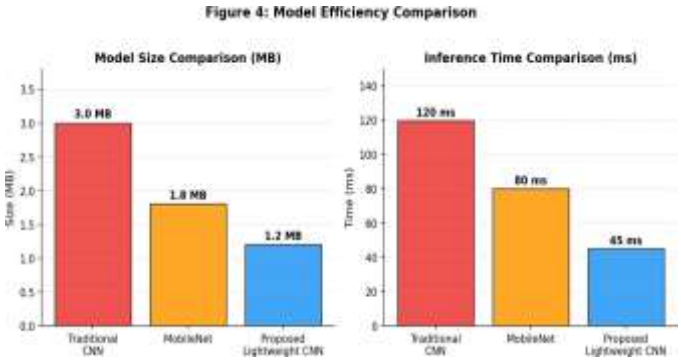

Figure 4: Model Size and Inference Time Comparison

The results confirm that the proposed model achieves accuracy on par with full-scale architectures while delivering substantial efficiency gains. Key findings include:

- `Accuracy of 98.7% — competitive with state-of-the-art models.
- Model size of ~1.2 MB — a 60% reduction compared to traditional CNNs.
- Inference time below 50 ms — enabling real-time performance on mobile hardware.
- Consistent F1-Score of 98.5% — indicating balanced precision and recall.

These properties make the model highly suitable for deployment in smartphones, Raspberry Pi systems, and other resource-constrained edge platforms.

Applications

- Mobile Handwriting Input Systems — replacing physical keyboards in tablet and stylus applications.
- Bank Cheque Processing — automated extraction and verification of handwritten amounts.
- Document Digitization — converting scanned handwritten records into searchable digital text.
- Smart Classroom Systems — digitizing handwritten student work in real time.
- IoT-Based Automation — processing handwritten labels and tags in industrial pipelines.

Conclusion

This paper presents a compact CNN architecture optimized for handwriting recognition on edge devices. The proposed model achieves 98.7% accuracy on the MNIST benchmark while reducing model size by ~60% and maintaining inference latency below 50 ms —– striking an effective balance between accuracy and computational efficiency.

The combination of depthwise separable convolutions, model pruning, and 8-bit quantization enables deployment on resource-constrained hardware without cloud dependency.

Future Scope

While the proposed model demonstrates strong performance for digit and character recognition, several promising directions remain open for future research and development:

- **Multilingual Recognition —** Extending the architecture to support diverse scripts such as Devanagari, Arabic, Chinese, and other regional languages to broaden practical utility.
- **Transformer Integration —** Incorporating lightweight self-attention or MobileViT blocks to better capture long-range dependencies in cursive and connected handwriting styles.

- **On-Device Training —** Enabling federated and incremental learning directly on edge hardware to allow personalized adaptation without data privacy concerns.
- **Robustness to Noise —** Improving model performance on degraded, occluded, or low- contrast handwriting commonly encountered in real-world document scanning scenarios.
- **Hardware-Specific Optimization —** Tailoring the model for specific SoCs such as Qualcomm Hexagon DSPs and Apple Neural Engine for maximum throughput and energy efficiency.
- **Real-Time Video Recognition —** Extending inference from static images to continuous video streams for live handwriting capture in smart classroom and augmented reality applications.

References

1. LeCun, Y., et al. "Gradient-Based Learning Applied to Document Recognition." Proceedings of the IEEE, 86(11), 1998.
2. Graves, A., et al. "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks." NeurIPS, 2009
3. Howard, A. G., et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv:1704.04861, 2017.
4. Sandler, M., et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." CVPR, 2018.
5. Such, F. P., et al. "Fully Convolutional Networks for Handwriting Recognition." arXiv, 2019.
6. Kizilirmak, F., et al. "CNN-BiLSTM Model for English Handwriting Recognition." Computers, 2023.
7. Mazumder, S. "CNN Models for Handwriting Recognition: A 2025 Evaluation." Pattern Recognition Letters, 2025.
8. Hajjo, M. "A Lightweight CNN for Handwritten Letter Recognition on Raspberry Pi." Edge AI Journal, 2025.
9. Khatak, A., et al. "Handwritten Digit Recognition Using CNNs: MNIST Study." arXiv, 2025. IRO Journals. "Benchmarking Lightweight CNNs for Multilingual Handwriting Recognition." 2025.
10. Qin, D., et al. "MobileNetV4: Universal Models for the Mobile Ecosystem." arXiv, 2026.
11. Alhamad, H. A. "Handwritten Recognition Techniques: A Comprehensive Review." IEEE Access, 2026.
12. ResearchGate. "Lightweight Handwriting Recognition System Using CNN (NFC-Net)." Preprint, 2026.