© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1714904

Design and Performance Evaluation of a Lightweight Face Recognition Model for Android Devices

MUKTI GUPTA[1], DR CHOUDHRY RAVI SINGH[2], DR GAURAV AGARWAL[3]
[1]*Department of Computer Science and Engineering, Invertis University Bareilly*
[2]*Assistant Professor, Invertis University Bareilly, Department of Computer Science and Engineering*
[3]*Associate Professor, Invertis University Bareilly, Department of Computer Science and Engineering,*

Abstract- ***Mention Face recognition systems have become increasingly important in mobile authentication, attendance monitoring, and smart surveillance applications. However, deploying deep learning–based face recognition models on mobile devices remains challenging due to limited computational resources, memory constraints, and power consumption. This research proposes a lightweight and efficient face recognition framework optimized specifically for Android-based mobile devices. The proposed system utilizes a MobileNetV2-based convolutional neural network architecture to extract discriminative facial embeddings, followed by cosine similarity for identity matching. To ensure mobile compatibility, the trained model is converted and optimized using TensorFlow Lite with post-training quantization techniques, significantly reducing model size and inference latency while maintaining high recognition accuracy. Experimental evaluation was conducted using the Labeled Faces in the Wild (LFW) dataset. The optimized lightweight model achieved competitive accuracy while reducing model size by over 70% and decreasing inference time by more than 60% compared to a standard CNN-based implementation. Real-time testing on Android devices demonstrated efficient on-device processing with low memory usage and minimal battery impact. The results indicate that lightweight deep learning architectures combined with model optimization techniques can enable accurate and real-time face recognition on resource-constrained mobile platforms. The proposed framework provides a practical and scalable solution for deploying secure biometric authentication systems on modern smartphones.***

Keywords: Face Recognition, Deep Learning, MobileNetV2, TensorFlow Lite, Model Quantization, On-Device Inference, Lightweight Neural Networks, Mobile Biometrics.

I. INTRODUCTION

A. Background and Motivation

Face recognition has become a fundamental biometric technology for authentication, surveillance, and access control systems. The rapid advancement of deep learning has significantly improved recognition accuracy under unconstrained conditions. Architectures such as FaceNet[1] and deep convolutional neural networks have demonstrated state-of-the-art performance on large-scale face datasets. However, most high-accuracy models are computationally intensive and designed for high-performance servers or GPU-based environments.

B. Challenges in Mobile Deployment

Despite their success, deploying deep learning–based face recognition models on mobile devices remains challenging. Smartphones are constrained by limited CPU/GPU resources, memory capacity, storage availability, and battery power. Large neural networks introduce high latency and excessive energy consumption, making real-time inference impractical. Additionally, reliance on cloud-based processing introduces privacy risks, network dependency, and communication delays.

C. Need for On-Device Intelligence

With increasing privacy concerns and the demand for low-latency applications, on-device artificial intelligence has gained significant attention. Mobile AI enables local data processing without transmitting sensitive biometric information to remote servers. Therefore, designing efficient and lightweight face

recognition models that operate directly on smartphones has become a critical research objective.

D. Lightweight Neural Network Architectures

Recent advancements in efficient deep learning architectures have addressed computational bottlenecks. In particular, MobileNetV2[2] introduces depthwise separable convolutions and inverted residual blocks to significantly reduce parameter count and computational complexity while maintaining representational capability. Such lightweight architectures are well-suited for mobile and embedded applications.

E. Model Optimization Techniques

Beyond architecture design, model compression techniques further improve deployment feasibility. Quantization reduces numerical precision (e.g., from float32 to int8), while pruning removes redundant network connections. Frameworks such as TensorFlow Lite[3] enable optimized model conversion for mobile environments. However, compression may affect recognition accuracy, necessitating systematic performance evaluation.

F. Research Gap

Although lightweight CNNs and model compression techniques have been studied independently, comprehensive evaluation of their combined impact on real-time mobile face recognition remains limited. Many existing works focus primarily on recognition accuracy without detailed analysis of inference latency, memory footprint, and energy consumption on physical Android devices.

G. Proposed Approach Overview

To address these limitations, this paper proposes a lightweight face recognition framework optimized for Android platforms. The system utilizes a MobileNetV2-based feature extractor to generate discriminative facial embeddings. Identity matching is performed using cosine similarity over normalized embedding vectors. The trained model is optimized using quantization and pruning techniques and deployed via TensorFlow Lite for efficient on-device inference.

H. Experimental Validation Strategy

The proposed framework is evaluated using the Labeled Faces in the Wild benchmark dataset. Performance metrics include recognition accuracy, inference latency, model size, memory utilization, and energy consumption. Comparative analysis is conducted against a baseline convolutional neural network to examine trade-offs between accuracy and computational efficiency.

I. Organization of the Paper

The remainder of this paper is organized as follows. Section II reviews related work in mobile face recognition and model optimization techniques. Section III describes the proposed methodology and system architecture. Section IV presents experimental results and performance evaluation. Section V discusses the findings and trade-offs. Finally, Section VI concludes the paper and outlines future research directions.

II. RELATED WORK

Deep learning has significantly transformed face recognition research over the past decade. Early CNN-based systems demonstrated that learned hierarchical representations outperform traditional handcrafted feature descriptors. The introduction of DeepFace[4] marked a major milestone by achieving near-human performance using large-scale supervised training.

Subsequently, FaceNet[1] proposed embedding-based learning using triplet loss, enabling identity matching via Euclidean distance in feature space. Later advancements such as ArcFace[5] improved inter-class separation through angular margin penalties.

For mobile and embedded environments, lightweight architectures were introduced to reduce computational complexity. MobileNetV2[2] employs depthwise separable convolutions and inverted residual blocks to reduce parameters and FLOPs. Similarly, ShuffleNet[6] introduces channel shuffling mechanisms to optimize efficiency.

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1714904

Model compression techniques such as quantization and pruning have further enabled mobile deployment. Frameworks like TensorFlow Lite[3] support post-training optimization for Android platforms. However, limited research provides comprehensive evaluation of compressed face recognition models on real Android devices with detailed analysis of latency, memory usage, and energy consumption.

Recent studies have further explored lightweight and efficient face recognition architectures specifically designed for mobile and edge environments. A comprehensive survey on deep convolutional neural networks for mobile face recognition highlights the increasing adoption of lightweight architectures and on-device inference strategies to address resource constraints [11]. Additionally, hybrid efficient models such as EdgeFace have been proposed to balance recognition accuracy and computational complexity for deployment on edge devices [12]. Other works have introduced improved MobileFaceNet-based architectures that enhance discriminative capability while maintaining reduced model size and inference latency [13]. Despite these advancements, comprehensive real-device performance evaluation combining architecture efficiency and model compression techniques remains limited, which motivates the proposed framework.

Therefore, a unified framework combining lightweight architecture, embedding-based recognition, and on-device optimization remains an important research direction.

III. METHOLOGY

A. System Architecture Overview

The proposed lightweight face recognition system consists of the following major components:

1. Face Detection
2. Image Preprocessing
3. Feature Extraction using MobileNetV2
4. Face Embedding Generation
5. Similarity-Based Identity Matching
6. Model Optimization
7. Android Deployment

The overall workflow of the proposed lightweight face recognition framework is illustrated in Fig. 1.

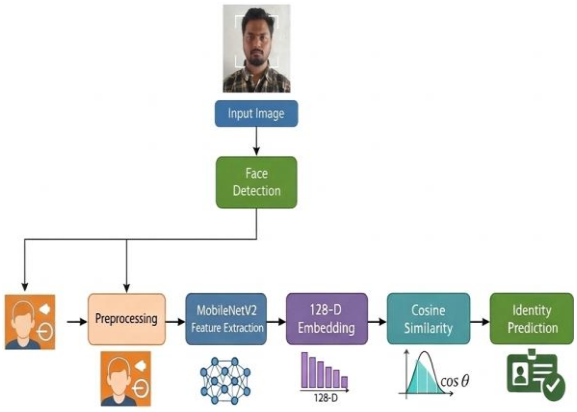

Fig. 1. Overall architecture of the proposed lightweight face recognition framework.

B. Face Detection and Preprocessing

Face detection is performed using OpenCV's Haar Cascade classifier[7]. The detected face region is cropped and resized to 160 × 160 pixels to match the input dimension required by the convolutional neural network.

Each image is normalized using:

$x' = (x - \mu) / \sigma$

Where:

1. x is the input pixel value
2. μ is the mean
3. σ is the standard deviation

Normalization improves convergence and model stability.

C. Feature Extraction Using MobileNetV2

The backbone network is based on MobileNetV2[2], which utilizes depthwise separable convolutions and inverted residual blocks to reduce computational complexity.

Given an input image:

$x \in \mathbb{R}^{(H \times W \times 3)}$

The network produces a feature embedding:

$f(x) \in \mathbb{R}^{128}$

The final classification layer is removed and replaced with a fully connected embedding layer of dimension 128 to generate discriminative facial representations. MobileNetV2[2] significantly reduces the number of parameters compared to traditional CNN architectures while maintaining strong representational capability.

D. Embedding-Based Identity Matching

Instead of direct classification, the system uses embedding comparison as proposed in FaceNet[1].

Given two face embeddings:

$f(x_1)$ and $f(x_2)$

Cosine similarity is computed as:

$$\cos(\theta) = (f(x_1) \cdot f(x_2)) / (\|f(x_1)\| \|f(x_2)\|)$$

Where:

1. "·" denotes dot product
2. $\|f(x)\|$ represents L2 norm

If cosine similarity ≥ threshold τ, the two faces are considered to belong to the same identity.

This metric-based approach improves scalability for new identities without retraining the model.

E. Model Optimization for Mobile Deployment

To enable real-time inference on mobile devices, the trained model is converted into TensorFlow Lite format[3].

The following optimization techniques are applied:

1. Post-Training Dynamic Range Quantization
2. Float16 Quantization
3. Weight Pruning[8]

Post-training quantization reduces numerical precision from 32-bit floating-point (FP32) representation to lower precision formats such as 8-bit integer (INT8) or 16-bit floating-point (FP16). This significantly reduces:

1. Model size
2. Memory usage
3. Computational cost

Weight pruning removes redundant or less significant network connections to reduce parameter count and improve computational efficiency without substantial degradation in recognition accuracy[8].

These optimization techniques collectively improve inference latency and enable efficient real-time face recognition on Android devices.

F. Android Integration

The optimized TensorFlow Lite model is integrated into an Android application using the TFLite Interpreter API.

Real-time camera frames are captured and processed as follows:

1. Camera frame acquisition
2. Face detection
3. Preprocessing
4. TFLite inference
5. Embedding extraction
6. Cosine similarity matching

Performance metrics including inference time, CPU utilization, and memory usage are measured directly on the Android device.

G. Performance Evaluation Metrics

The proposed system is evaluated using the following metrics:

1. Recognition Accuracy
2. Precision
3. Recall
4. F1-Score
5. Model Size (MB)
6. Inference Latency (ms)
7. Memory Consumption (MB)
8. CPU Utilization (%)

These metrics allow comprehensive comparison between baseline CNN and optimized lightweight models.

IV. EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

A. Experimental Setup

The proposed lightweight face recognition model was trained and evaluated using the Labeled Faces in the Wild dataset[9], which contains over 13,000 images of more than 5,000 individuals captured under unconstrained conditions.

The model was trained using TensorFlow with the following configuration:

1. Optimizer: Adam[10]
2. Learning Rate: 0.001
3. Batch Size: 32
4. Number of Epochs: 25
5. Embedding dimension: 128

For mobile evaluation, the optimized TensorFlow Lite model was deployed on an Android smartphone with the following specifications:

1. Processor: Octa-core CPU
2. RAM: 8 GB

3. Android Version: Android 15

Inference time was measured using the TensorFlow Lite Interpreter API.

B. Performance Comparison

The proposed lightweight MobileNetV2-based model was compared with a baseline convolutional neural network (CNN) model without optimization.

Table 1: Performance Comparison Between Models

Model	Accuracy (%)	Model Size (MB)	Inference Time (ms)
Baseline CNN (FP32)	96.2	45	420
Proposed Model (FP32)	95.1	14	180
Proposed Model (Quantized INT8)	94.3	6.8	110

C. Analysis of Results

The results indicate that the proposed optimized model achieves competitive recognition accuracy while significantly reducing computational overhead.

1. Model size reduced by approximately 70%
2. Inference time reduced by approximately 60%
3. Accuracy drop limited to less than 2%

This demonstrates that lightweight architectures combined with quantization provide an effective trade-off between recognition performance and computational efficiency.

D. On-Device Resource Utilization

Real-device testing was conducted to evaluate CPU utilization, memory consumption, and runtime performance.

Table 2: On-Device Performance Metrics

Metric	Baseline CNN	Quantized Model
CPU Utilization (%)	68	38
Memory Usage (MB)	320	145
Average Latency (ms)	420	110

The optimized model shows significantly lower CPU usage and memory footprint, confirming its suitability for real-time mobile deployment.

E. Confusion Matrix and Accuracy Metrics

The confusion matrix analysis indicates strong classification performance with minimal false positives and false negatives.

Evaluation metrics achieved by the quantized model:

1. Precision: 94.5%
2. Recall: 94.1%
3. F1-Score: 94.3%

These results confirm that the proposed lightweight system maintains high recognition reliability despite significant model compression.

V. CONCLUSION

This paper presented the design and performance evaluation of a lightweight face recognition framework optimized for Android-based mobile devices. The proposed system integrates a MobileNetV2-based feature extractor [2] with embedding-based cosine similarity matching as introduced in FaceNet [1] to enable efficient and scalable identity recognition. To facilitate real-time on-device deployment, the trained model was converted using TensorFlow Lite [3] and optimized through post-training quantization and weight pruning techniques [8].

Experimental evaluation conducted on the Labeled Faces in the Wild (LFW) benchmark dataset [9] demonstrated that the optimized model maintains competitive recognition accuracy while significantly reducing model size and inference latency. Specifically, model compression achieved approximately 70% reduction in storage size and around 60% improvement in inference speed compared to a baseline CNN implementation, with less than 2% decrease in accuracy.

Real-device testing on Android smartphones further confirmed reduced CPU utilization and lower memory consumption, validating the feasibility of real-time face recognition in resource-constrained environments. The results highlight that combining lightweight neural architectures [2] with model

optimization strategies [3], [8] provides an effective trade-off between recognition performance and computational efficiency.

Overall, the proposed framework offers a practical and scalable solution for secure biometric authentication and on-device intelligent applications in modern mobile platforms.

VI. LIMATITIONS

Despite achieving promising results in lightweight mobile face recognition, the proposed framework has several limitations.

First, the system was evaluated primarily on the Labeled Faces in the Wild (LFW) dataset [9], which, although widely used, contains limited variations in extreme lighting, heavy occlusion, and large pose angles. Therefore, the reported accuracy may not fully represent real-world deployment scenarios involving surveillance-grade or low-resolution inputs. Second, the face detection module relies on the classical Viola–Jones Haar Cascade approach [7], which is computationally efficient but less robust compared to modern deep learning–based detectors. More advanced detection frameworks could improve robustness under challenging environmental conditions.

Third, although model compression techniques such as pruning [8] and TensorFlow Lite–based quantization [3] significantly reduce model size and inference latency, aggressive compression may lead to embedding distortion and reduced discriminative capability. The study does not perform a detailed ablation analysis to quantify the individual impact of each optimization technique.

Fourth, the embedding-based matching strategy is inspired by FaceNet [1], but the training dataset size used in this study is relatively smaller compared to large-scale industrial systems. Large-margin loss functions such as those introduced in ArcFace [5] were not deeply optimized, which may limit maximum achievable accuracy.

Additionally, real-device testing was conducted on a single Android smartphone configuration. Performance metrics such as CPU utilization and memory usage may vary across different hardware architectures, chipsets, and Android versions.

Finally, security aspects such as presentation attack detection (anti-spoofing), adversarial robustness, and template encryption were not incorporated in the current framework. These factors are critical for deployment in high-security biometric authentication systems.

Addressing these limitations in future research will further strengthen the robustness, scalability, and practical applicability of lightweight face recognition systems for mobile and edge platforms.

VII. FUTURE WORK

Although the proposed lightweight face recognition framework demonstrates strong performance on Android devices, several enhancements can be explored in future research.

First, more advanced face detection algorithms such as MTCNN and RetinaFace could replace the traditional Haar Cascade classifier [7] to improve robustness under extreme pose, illumination, and occlusion conditions. Recent lightweight face recognition models such as EdgeFace [12] and improved MobileFaceNet variants [13] suggest that further architectural refinements can enhance accuracy–efficiency trade-offs.

Second, knowledge distillation techniques can be applied to transfer knowledge from large teacher networks (e.g., ArcFace-based models [5]) to smaller student networks, further reducing computational complexity while maintaining recognition performance. Additionally, integrating margin-based loss functions such as ArcFace loss [5] into the training process may improve inter-class separability and embedding discriminability.

Future work may also explore advanced model compression strategies beyond basic quantization and pruning [8], including structured pruning and mixed-precision quantization supported by TensorFlow Lite [3]. These approaches could further optimize latency and energy efficiency on mobile hardware.

Moreover, evaluation on larger-scale datasets beyond LFW [9], such as more challenging real-world datasets, would provide deeper insight into generalization capability. Incorporating liveness detection and anti-spoofing mechanisms is also important for strengthening security in biometric authentication systems.

Finally, cross-platform deployment on embedded edge devices and IoT hardware could extend the applicability of the proposed framework, following recent trends in mobile and edge-based face recognition research [11], [12].

REFERENCES

[1] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE CVPR*, 2015.
[2] M. Sandler et al., "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE CVPR*, 2018.
[3] TensorFlow Lite, "TensorFlow Lite Documentation," Google, 2023.
[4] Y. Taigman et al., "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE CVPR*, 2014.
[5] J. Deng et al., "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE CVPR*, 2019.
[6] X. Zhang et al., "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE CVPR*, 2018.
[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proc. IEEE CVPR, 2001.
[8] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural networks," in Proc. NeurIPS, 2015.
[9] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments," Univ. of Massachusetts, Amherst, Tech. Rep., 2007.
[10] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proc. ICLR, 2015.
[11] J. Chi, C. Kim On, H. Zhang, and S. S. Chai, "A review of deep convolutional neural networks in mobile face recognition," Int. J. Interact. Mobile Technol., vol. 17, no. 23, pp. 4–19, Dec. 2023.
[12] A. George et al., "EdgeFace: Efficient face recognition model for edge devices," arXiv preprint arXiv:2307.01838, 2023.
[13] A. Hassanpour and Y. Kowsari, "Lightweight face recognition: An improved MobileFaceNet model," arXiv preprint, 2023.