

Lightweight Retinal Disease Classification for Offline Mobile Screening: Model Selection, TensorFlow Lite Optimization, and Flutter Deployment

Manya Jamil Farah ^{a,*}, Dr. Bassel Mustaffa Alkhatib ^a

^a Web Science program, Syrian Virtual University, Damascus, Syria

Corresponding Author : Manya Jamil Farah

Supervisor: Dr. Bassel Mustaffa Alkhatib

Abstract

Early detection of retinal diseases such as cataract, diabetic retinopathy, and glaucoma is essential to prevent irreversible vision loss, particularly in low-resource regions where access to specialists and reliable internet connectivity may be limited. This study presents an end-to-end pipeline for lightweight retinal disease classification and fully offline mobile deployment. Two efficient convolutional neural networks, MobileNetV2 and EfficientNetB0, were evaluated on a public Kaggle fundus image dataset containing four classes (Cataract, Diabetic Retinopathy, Glaucoma, and Normal). Under identical training conditions, MobileNetV2 achieved superior performance, reaching 91.52% test accuracy with only 2.26 million parameters, outperforming EfficientNetB0 (90.74%). Several TensorFlow Lite post-training optimization techniques were investigated, including FP16 quantization, dynamic-range INT8, full-integer INT8, pruning, and knowledge distillation. Among these, FP16 quantization provided the best balance between accuracy, model size, and latency, reducing the model size from 20.76 MB to 4.26 MB while preserving accuracy (91.99%). The optimized model was integrated into a Flutter-based Android application capable of fully offline inference. Latency benchmarks on multiple mid-range smartphones (128–316 ms per image) demonstrate the feasibility of real-time screening. Although evaluated on a relatively small public dataset, the proposed pipeline demonstrates that carefully optimized lightweight CNNs can enable practical and privacy-preserving retinal disease screening directly on commodity smartphones.

Keywords Eye disease classification; fundus images; MobileNetV2; EfficientNetB0; TensorFlow Lite; model quantization; on-device inference; Flutter; mobile health; resource-constrained environments.

1. Introduction

Retinal diseases such as cataract, diabetic retinopathy (DR), age-related macular degeneration (AMD), and glaucoma remain leading causes of preventable visual impairment worldwide [1] [2] [3] [4]. According to the World Health Organization, hundreds of millions of people are affected, and a substantial fraction of these cases could be avoided through timely detection and treatment [5]. However, in many low- and middle-income regions, systematic screening is hindered by limited access to ophthalmologists, diagnostic imaging infrastructure, and reliable internet connectivity. This context motivates the development of automated, low-cost, and portable screening tools that can operate directly on commodity mobile devices.

Deep convolutional neural networks (CNNs) have demonstrated expert-level performance in detecting retinal diseases from color fundus photographs [6] [7] [8] [9]. Many of these systems are built on large architectures such as Inception, ResNet, or DenseNet and are typically deployed in cloud or data-center settings. While such setups achieve high accuracy, they are not always feasible for rural clinics, outreach screening programs, or offline use, where latency, privacy, and hardware limitations become critical.

To address this gap, recent research has shifted toward lightweight CNN architectures that balance diagnostic accuracy with computational efficiency. MobileNetV2 [10] and EfficientNetB0 [11] are two

prominent examples. However, systematic comparisons of these models under identical conditions for multi-class eye disease classification remain limited. Moreover, while several studies report high accuracy using heavy or ensemble models, they often neglect the practical aspects of on-device deployment, such as quantization, model size reduction, and real-world latency benchmarks.

This paper bridges the gap between model-centric research and deployment reality by presenting a complete end-to-end pipeline:

We first compare MobileNetV2 and EfficientNetB0 on a public Kaggle fundus dataset [12] using a unified training and evaluation protocol, identifying MobileNetV2 as the best trade-off between accuracy and efficiency.

We then apply post-training optimization techniques—FP16 quantization, dynamic-range INT8, full-integer INT8, pruning, and knowledge distillation—to the selected MobileNetV2 model, evaluating their impact on accuracy, model size, and inference speed.

Finally, we integrate the best-performing variant (FP16) into a Flutter-based Android application, benchmark its latency on multiple mid-range devices, and verify consistency between cloud and on-device inference.

The key contributions of this work are threefold:

1. A controlled experimental comparison between two lightweight CNN architectures (MobileNetV2 and EfficientNetB0) for multi-class retinal disease classification under identical training conditions.
2. A systematic evaluation of multiple TensorFlow Lite optimization strategies (FP16, dynamic-range INT8, full-integer INT8, pruning, and knowledge distillation) with respect to accuracy, model size, and CPU inference latency.
3. A complete end-to-end mobile deployment pipeline, including a Flutter-based offline Android application and real-device latency benchmarking across multiple mid-range smartphones.

The novelty of this work lies not in proposing a new CNN architecture, but in demonstrating a complete and practically deployable pipeline that connects lightweight model selection, systematic TensorFlow Lite optimization, and fully offline smartphone deployment for retinal disease screening.

2. Background and Related Work

2.1. Lightweight CNNs for Retinal Disease Classification

CNN-based systems have been widely applied to retinal imaging [6] [7] [8] [9]. Early works relied on large backbones (Inception, ResNet, DenseNet) achieving high accuracy but requiring substantial computational resources. In parallel, lightweight architectures such as MobileNet [13], MobileNetV2 [10], and ShuffleNet [14] were developed for general-purpose computer vision using depthwise separable convolutions and inverted residuals. Recent studies have begun applying these families to retinal disease classification [15] [16] [17] [18], showing that lightweight models can maintain clinically meaningful accuracy while drastically reducing parameter counts.

Table 1 summarizes key prior studies. Most achieve high accuracy but either use heavy architectures or do not address on-device deployment. Our work explicitly focuses on the intersection of accuracy, efficiency, and real-world mobile deployment.

Table 1 Summary of key prior studies on retinal disease classification.

Study (Year)	Dataset Type	Model / Algorithm	Diseases / Tasks	Key Findings
Gulshan et al., 2016 [7]	Fundus (128k)	Inception-v3	DR	Sensitivity 97.5%, Specificity 93.4%
Ting et al., 2017 [8]	Multi-ethnic Fundus	Inception-v4 multi-task	DR, AMD, Glaucoma	Sensitivity 90.5%, Specificity 91.6%

Kermany et al., 2018 [6]	OCT & Fundus	Inception-v3 Transfer	CNV, DME, Drusen, Normal	Accuracy 96.4–96.5%
Fu et al., 2018 [19]	Real Fundus	Ensemble CNN	Glaucoma	Accuracy >95%
Sahlsten et al., 2019 [20]	Real Fundus	ResNet-50	DR	AUC 0.98
Arnob et al., 2025 [15]	Fundus	Lightweight CNN + XAI	4 diseases	Accuracy 95.8%, size 6MB
Zafar et al., 2025 [17]	EyePACS	MobileNetV2	DR (5-grade)	Accuracy 92.7%, 120ms latency
Wang et al., 2024 [16]	Fundus	Many-MobileNet	4 diseases	Accuracy 97%, medium size

Most prior studies focus primarily on predictive accuracy using large CNN architectures or ensemble models. While such approaches can achieve very high classification performance, they often require powerful hardware and cloud-based deployment. In contrast, the present work emphasizes deployability and efficiency, demonstrating that a lightweight architecture can achieve competitive performance while remaining suitable for fully offline execution on commodity smartphones.

2.2. On-Device Machine Learning and TensorFlow Lite

Frameworks like TensorFlow Lite, Core ML, and ONNX Runtime enable efficient inference on mobile devices. TensorFlow Lite supports post-training quantization to FP16 and INT8, reducing model size and latency with minimal accuracy loss [16] [10]. FP16 quantization stores weights in half-precision, typically preserving accuracy while cutting model size by half. INT8 variants achieve greater compression but may require calibration and can degrade accuracy.

In medical imaging, on-device inference offers advantages: offline operation, lower latency, improved privacy, and suitability for low-resource settings. Several works have explored mobile deployment of CNN-based medical models [15] [17], but many lack rigorous comparison of optimization techniques or real-device benchmarks.

2.3. Gap and Positioning

Despite progress, a gap remains between model-centric studies (high accuracy, no deployment) and system-level studies (apps, but limited optimization comparison). This work fills that gap by combining a controlled model comparison with systematic optimization and real-device deployment, using the same dataset and evaluation protocol throughout.

3. Materials and Methods

3.1. Dataset and Preprocessing

We use a public Kaggle dataset [12] containing 4,217 color fundus images labeled into four classes: Cataract, Diabetic Retinopathy, Glaucoma, and Normal. The dataset is split stratified by class into 70% training (2,949 images), 15% validation (631), and 15% test (637). Table 2 shows the distribution. All experiments, dataset splits, and model configurations were kept fully consistent across both backbones to ensure methodological fairness and experimental reproducibility.

Figure 1 shows the sample fundus images from the four classes.

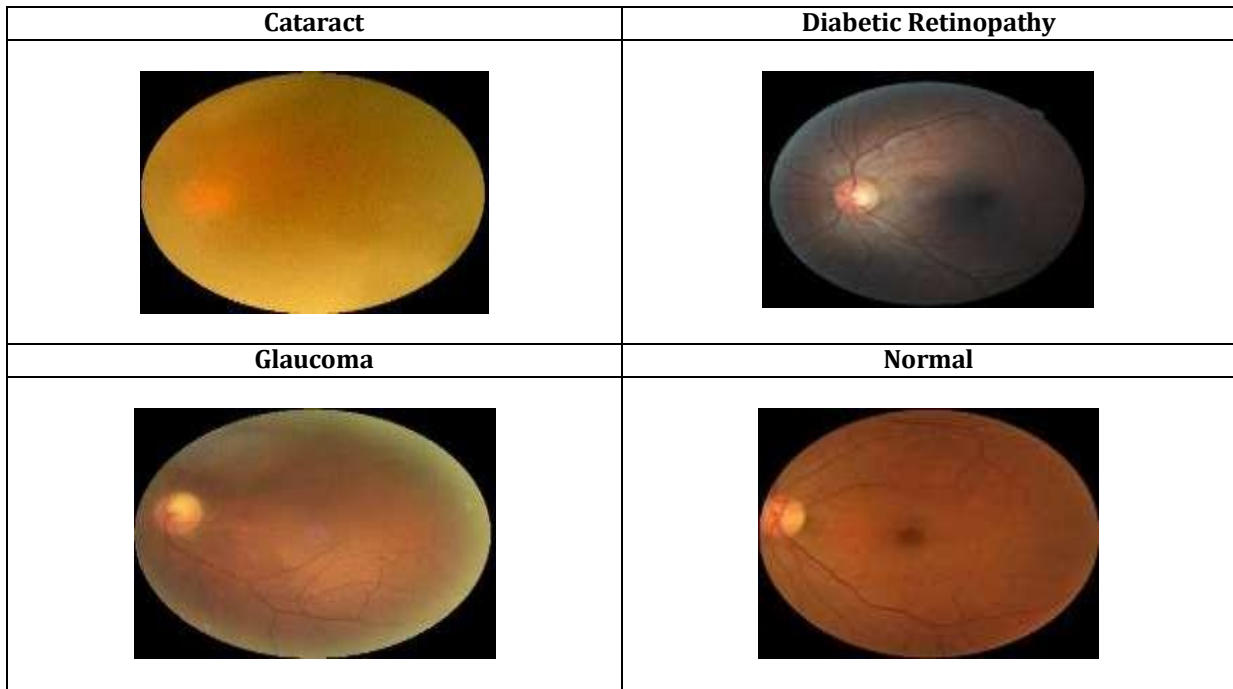


Figure 1 Sample fundus images from the four classes

Table 2 Number of images per class and per split (train/validation/test).

Class	Train	Validation	Test	Total
Cataract	726	155	157	1038
Diabetic Retinopathy	768	164	166	1098
Glaucoma	704	151	152	1007
Normal	751	161	162	1074
Total	2949	631	637	4217

All images are resized to 224×224 pixels and normalized. During training, we apply conservative data augmentation: random brightness and contrast adjustments (± 0.05) to improve generalization without distorting medical features.

Figure 2 illustrates the complete pipeline of the proposed system, including dataset preparation, model training and evaluation, TensorFlow Lite optimization, and final mobile deployment.

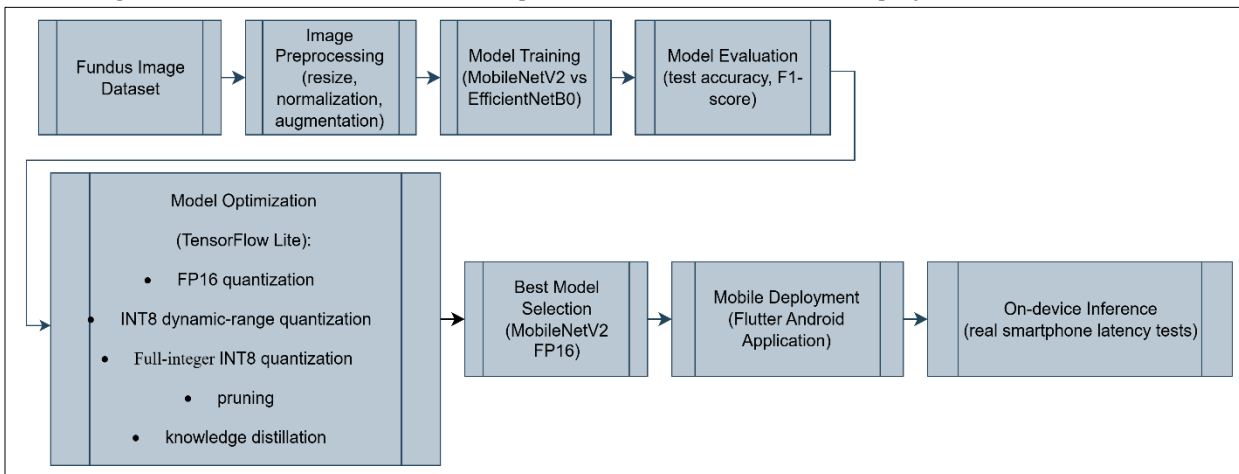


Figure 2 End-to-end pipeline of the proposed retinal disease classification and mobile deployment framework.

3.2. Model Selection: MobileNetV2 vs EfficientNetB0

We compare two lightweight backbones pre-trained on ImageNet: MobileNetV2 [10] and EfficientNetB0 [11]. Both are adapted by removing the top classification layer and adding a custom head: GlobalAveragePooling2D, Dropout, and a Dense layer with 4 units and softmax activation. The resulting models have approximately 2.26M parameters (MobileNetV2_Eye) and 4.05M parameters (EfficientNetB0_Eye).

Training follows a two-stage protocol:

- **Feature extraction:** Backbone frozen, head trained with Adam ($lr=1e-3$) for up to 50 epochs.
- **Fine-tuning:** Last 30 layers unfrozen, training continued with reduced learning rate (ReduceLROnPlateau, patience=2) and early stopping (patience=5). Batch size 32, loss sparse categorical Crossentropy.

Training experiments were conducted in the Google Colab environment, while inference latency measurements were performed on CPU to reflect the intended mobile deployment conditions.

These transformations are applied on the fly using Keras preprocessing layers, ensuring that each training epoch sees slightly different variations of the same underlying images.

Figure 3 shows the overall classification architecture used for both backbones

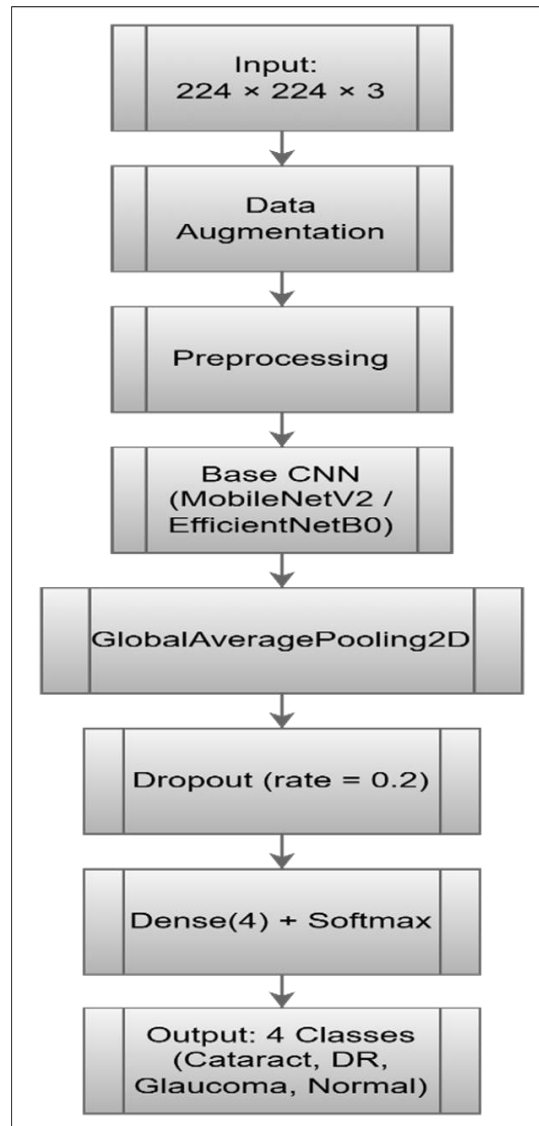


Figure 3 Overall classification architecture used for both backbones

Fundus images are resized to 224×224, optionally augmented, and passed through a shared preprocessing stage, then through either a MobileNetV2_Eye or EfficientNetB0 backbone, followed by a global average pooling layer, dropout, and a 4-unit softmax classifier.

3.3. Post-Training Optimization Techniques

From the trained MobileNetV2_Eye Keras model (baseline), we apply several TensorFlow Lite optimizations:

FP16 quantization: Weights cast to float16; inference uses float32 accumulators.

Dynamic-range INT8 quantization (DRQ): Weights quantized to int8, activations in float.

Full-integer INT8 quantization: Both weights and activations quantized to int8, requiring a representative calibration dataset.

Pruning: 50% unstructured sparsity applied to the Keras model before conversion.

Knowledge distillation (KD): A student (MobileNetV2) trained to mimic an EfficientNetB0 teacher. All variants are evaluated on the same test set to measure accuracy, model size, and (where applicable) inference latency in Colab (CPU). The best-performing variant is selected for mobile deployment.

3.4. Mobile Application Architecture and On-Device Inference

The FP16 TensorFlow Lite model is integrated into a Flutter-based Android application. The app allows image capture/selection, applies the same preprocessing as training (resize to 224×224, normalization), and invokes the TFLite interpreter for inference. Results are displayed as class probabilities. The app operates fully offline, with no cloud dependency.

3.5. Latency Benchmarking

Inference latency is measured on four platforms:

- Samsung Galaxy A55 (Exynos 1480, 8GB RAM)
- Redmi Note 11 (Snapdragon 680, 6GB RAM)
- Infinix Hot 20s (MediaTek Helio G96, 8GB RAM)
- Android Emulator (x86, mid-range host)

For each device, we run 10 repetitions per test image and report mean latency per sample (milliseconds).

4. Results

4.1. Model Selection Results

Table 3 shows overall test accuracy and model size for the two backbones. MobileNetV2_Eye achieves 91.52% accuracy, outperforming EfficientNetB0 (90.74%) despite having fewer parameters. The accuracy difference of 0.78 percentage points corresponds to about 5 additional correctly classified images out of 637 test samples.

Table 3 Overall performance of MobileNetV2_Eye vs. EfficientNetB0 on the test set.

Model	Test Accuracy (%)	Macro F1	Parameters (Millions)	Keras Size (MB)
MobileNetV2_Eye	91.52	0.91	2.26	20.76
EfficientNetB0	90.74	0.90	4.05	~35

Per-class metrics (Table 4

Table 5) reveal that glaucoma is the most challenging class for both models ($F1 \approx 0.83$), while diabetic retinopathy and cataract are well recognized. MobileNetV2 achieves higher recall for normal and glaucoma.

Table 4 Per-class precision, recall, and F1-score for MobileNetV2_Eye.

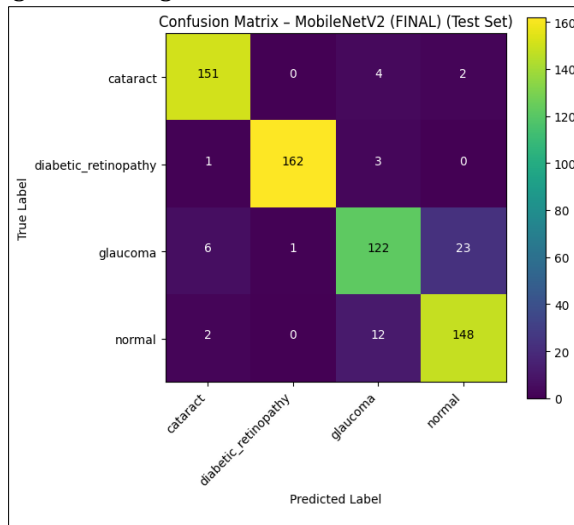
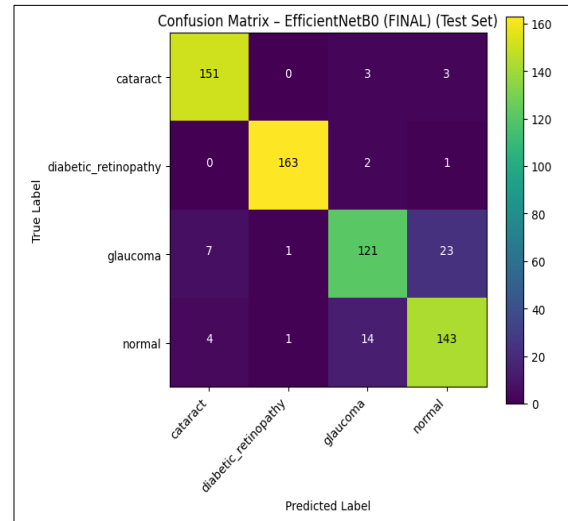
Class	Precision	Recall	F1-score	Support
Cataract	0.9437	0.9618	0.9527	157
Diabetic Retinopathy	0.9939	0.9759	0.9848	166
Glaucoma	0.8652	0.8026	0.8328	152
Normal	0.8555	0.9136	0.8836	162
Accuracy			0.9152	637
F1-weighted avg			≈ 0.9149	

Table 5 Per-class precision, recall, and F1-score for EfficientNetB0.

Class	Precision	Recall	F1-score	Support
Cataract	0.9321	0.9618	0.9467	157
Diabetic Retinopathy	0.9879	0.9819	0.9849	166
Glaucoma	0.8643	0.7961	0.8288	152
Normal	0.8412	0.8827	0.8614	162
Accuracy			0.9074	637
F1-weighted avg			≈ 0.9055	

Confusion matrices (Figure 4 and Figure 5) show that misclassifications mainly occur between glaucoma and normal, and occasionally between cataract and normal.

Figure 4 and Figure 5 show the confusion matrix on the test set.

**Figure 4 Confusion matrix of MobileNetV2_Eye on the test set****Figure 5 Confusion matrix of EfficientNetB0 on the test set**

Typical patterns observed:

- **Cataract and Normal** classes are generally well recognized by both models, with MobileNetV2_Eye achieving higher recall and fewer false negatives with normal class.
- **Diabetic Retinopathy** is also detected with good high recall, though some confusion with Normal persists, especially in borderline cases and low-quality images.
- **Glaucoma** is the most challenging class for both models, manifesting as reduced recall and frequent confusion with Normal and other disease classes. This observation is consistent with findings reported in previous studies in the literature: glaucomatous changes may be subtler and more heterogeneous than lesions associated with DR or dense cataract.

Misclassification in the glaucoma class is mainly attributed to the subtle visual progression of optic-nerve damage and the lack of clear macroscopic fundus patterns, which makes the class intrinsically harder to separate both clinically and algorithmically.

From a modeling perspective, the relatively lower recall for glaucoma suggests that purely 2D, RGB fundus images may not capture all discriminative cues required for reliable screening. Incorporating additional modalities (such as OCT or visual field measurements) or designing glaucoma-specific attention mechanisms could help the model focus on subtle optic nerve head changes. In Section 6, we therefore highlight glaucoma detection as a primary target for future methodological improvements. To further assess the discriminative performance of the final selected model, receiver operating characteristic (ROC) curves were computed for each class on the test set using the FP16 TensorFlow Lite model. As shown in Figure 6, the model achieved high AUC values across all four categories, with a macro-AUC of 0.981. The highest AUC was observed for diabetic retinopathy (0.999), followed by cataract (0.996), while glaucoma remained the most challenging class (0.956), which is consistent with the class-wise results reported earlier.

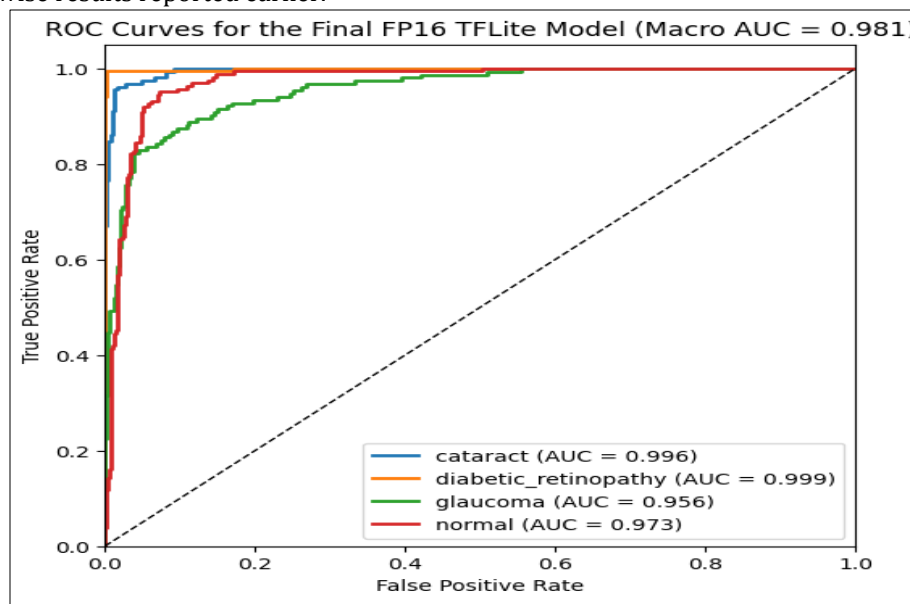


Figure 6 Receiver operating characteristic (ROC) curves for the four disease classes using the final FP16 TensorFlow Lite model on the test set.

4.2. Training Dynamics

The curves for the validation accuracy comparison during training between MobileNetV2_Eye model and EfficientNetB0 are shown in Figure 7.

And the curves for the validation loss comparison during training between MobileNetV2_Eye model and EfficientNetB0 are shown in Figure 8.

Experimental results showed that both models, MobileNetV2_Eye and EfficientNetB0, performed well on the validation set, exceeding 92% accuracy in the best settings for both. However, a slight decrease in performance accuracy was observed when evaluating the test set, with this decrease being more pronounced in the EfficientNetB0 model compared to the MobileNetV2_Eye model.

This difference can be explained by the fact that the validation set is used during training to make crucial decisions such as early stopping, adjusting the learning rate, and selecting the best version of the model, thus partially adapting the model to the characteristics of this set. In contrast, the test set remains completely isolated from the training process and is used solely to measure the model's ability to generalize to new, previously unseen data.

The larger parameter count of EfficientNetB0 may increase representational capacity, but it can also increase the risk of overfitting when the dataset size is limited.

This behavior indicates mild validation-induced overfitting in EfficientNetB0, likely caused by its larger capacity and higher parameter count, which led to weaker generalization on unseen test data compared to the more compact MobileNetV2_Eye model.

Conversely, the MobileNetV2_Eye model has a lighter and less complex architecture, making it less prone to overfitting to validation data and more stable in generalizing to the test set. This translates into slightly higher test accuracy than EfficientNetB0, along with balanced performance across precision, recall, and F1-score metrics, particularly in sensitive categories such as glaucoma.

Based on these results, the MobileNetV2_Eye model was adopted as the final model for comparison and subsequent development, given its superior test accuracy, stability in generalization, and computational lightness, which aligns with the study's objective of operating in resource-limited environments and being deployable on mobile devices.

Meanwhile, the results of EfficientNetB0 were retained as an important reference point demonstrating the impact of increasing model complexity on validation versus testing performance, and supporting comparative analysis between different lightweight models.

Figure 7 shows the validation accuracy comparison during training between MobileNetV2_Eye model and EfficientNetB0.

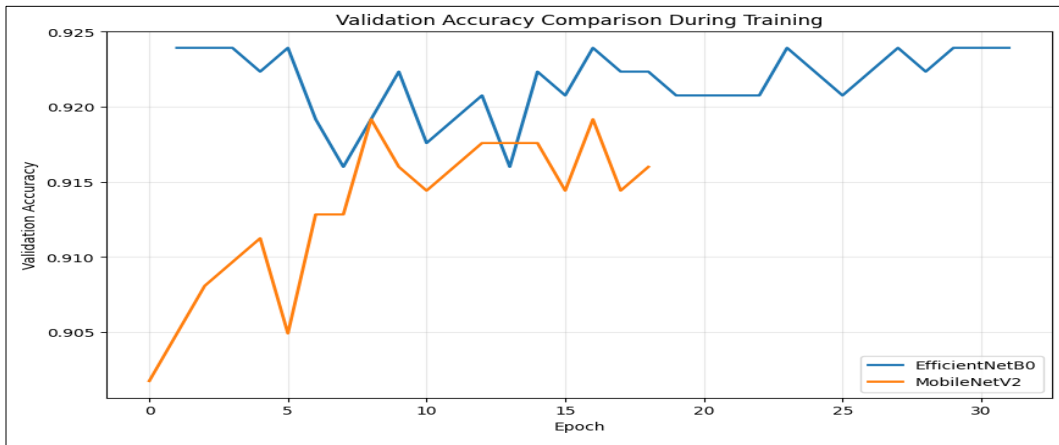


Figure 7 Validation accuracy comparison during training between MobileNetV2_Eye model and EfficientNetB0

Figure 8 shows the validation loss comparison during training between MobileNetV2_Eye model and EfficientNetB0.

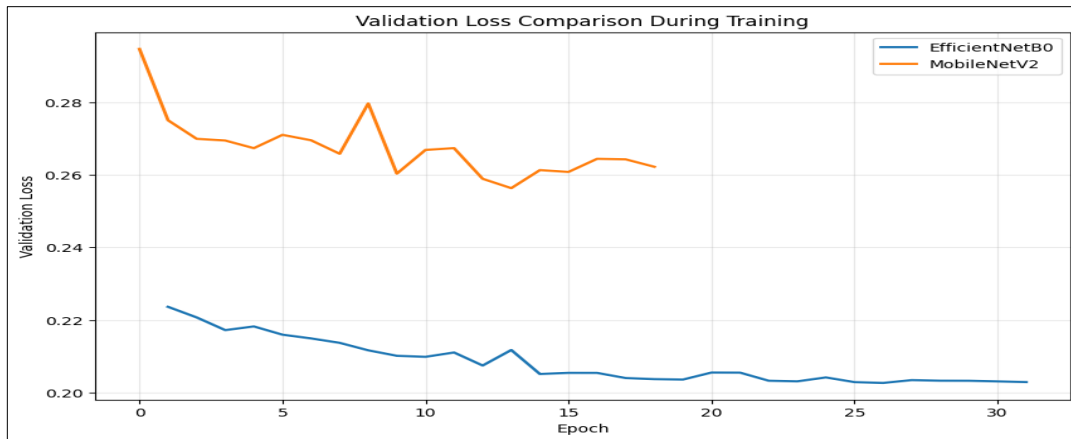


Figure 8 Validation loss comparison during training between MobileNetV2_Eye model and EfficientNetB0

These curves indicate that MobileNetV2 converges more stably and maintains slightly higher validation accuracy throughout training, suggesting improved generalization on this dataset. Figure 9 shows a randomly selected set of images from the test set, with the True Label (T) and the Predicted Label (P) indicated above each image. The labels for correct predictions are colored green, while incorrect predictions are highlighted in red.



Figure 9 Random examples from the test set with predictions from the MobileNetV2_Eye model

4.3. Comparison with Related Work

To position our results within the broader literature, we compare them against reported performances of eye disease classification systems using fundus images.

Table 6 Comparison of our MobileNetV2_Eye model with selected prior works (datasets, architectures, number of classes, and test accuracy)

Study	Dataset / Diseases	Backbone	No. Classes	Test Accuracy (%)	Approx. Params	Deployability
Gulshan et al., 2016 [7]	DR screening (EyePACS-like, 2-class)	Inception-v3	2	97.5	>25M	Cloud / high-end GPU

Ting et al., 2017 [8]	DR, AMD, Glaucoma (multi-ethnic)	Inception-v4 (MTL)	4	90.5	~40M	Data-center setting
Arnob et al., 2025 [15]	4 retinal diseases	Lightweight CNN	4	95.8	~6M	Potentially mobile
Wang et al. [16]	Cataract, DR, Glaucoma, Normal	Many-MobileNet	4	97.0	~5–6M × N	Lightweight but ensemble
This work	Cataract, DR, Glaucoma, Normal	MobileNetV2_Eye	4	91.99	~2.26M	4.26 MB TFLite model, mobile-ready

Table 6 shows that while our MobileNetV2_Eye model does not reach the highest accuracies reported for larger, non-deployable models, it achieves a competitive 91.99% accuracy with only ≈ 2.26 million parameters and a 4.262 MB TensorFlow Lite footprint. This design represents a deliberate and favorable trade-off, prioritizing deployability and efficiency on resource-constrained mobile hardware—the primary objective of this study—over marginal gains in absolute accuracy. The model is intentionally kept small for deployment on mid-range devices without cloud dependency.

4.4. Optimization Results

Table 7 compares the baseline Keras model with various TensorFlow Lite optimizations. FP16 quantization reduced the model size from 20.76 MB to 4.26 MB while preserving accuracy. A small numerical increase in the observed test accuracy was measured, which is likely within normal floating-point inference variance. Dynamic-range INT8 achieves further compression (2.39 MB) but does not improve accuracy compared with the baseline. Full-integer INT8 yields 2.59 MB but accuracy falls to 89.01%. Pruning and knowledge distillation (

Table 8) either degrade accuracy or offer no advantage over FP16.

Table 7 Baseline model vs. TensorFlow Lite quantized variants

Method	Technique	Test Acc	Size (MB)	Latency (ms/sample, CPU)
Baseline (Keras)	None	0.9152	20.762	302.199
TFLite FP16	Quantization	0.9199	4.262	12.143
TFLite DRQ (INT8)	Dynamic-range quantization	0.9152	2.392	22.620
TFLite INT8 (Full)	Full-integer INT8 quantization	0.8901	2.586	17.002

Table 8 Results of pruning experiments

Method	Technique	Test Acc
Pruned50 (Keras)	50% sparsity	0.8979
Pruned50 → TFLite FP16	Pruning + Quantization	0.8979
Pruned50 → TFLite DRQ	Pruning + Quantization	0.8932
KD Student (Keras)	Distilled from EfficientNet	0.9137
KD Student → TFLite FP16	Distillation + Quantization	0.9137
KD Student → TFLite DRQ	Distillation + Quantization	0.9073

FP16 offers the best trade-off: accuracy preservation, significant size reduction, and low latency. It is therefore selected for mobile deployment.

To better illustrate the trade-off between model compactness and predictive performance, Figure 10 presents a comparison between model size and test accuracy for the evaluated model variants. The results show that FP16 quantization significantly reduces model size while maintaining almost identical classification performance compared with the original Keras model.

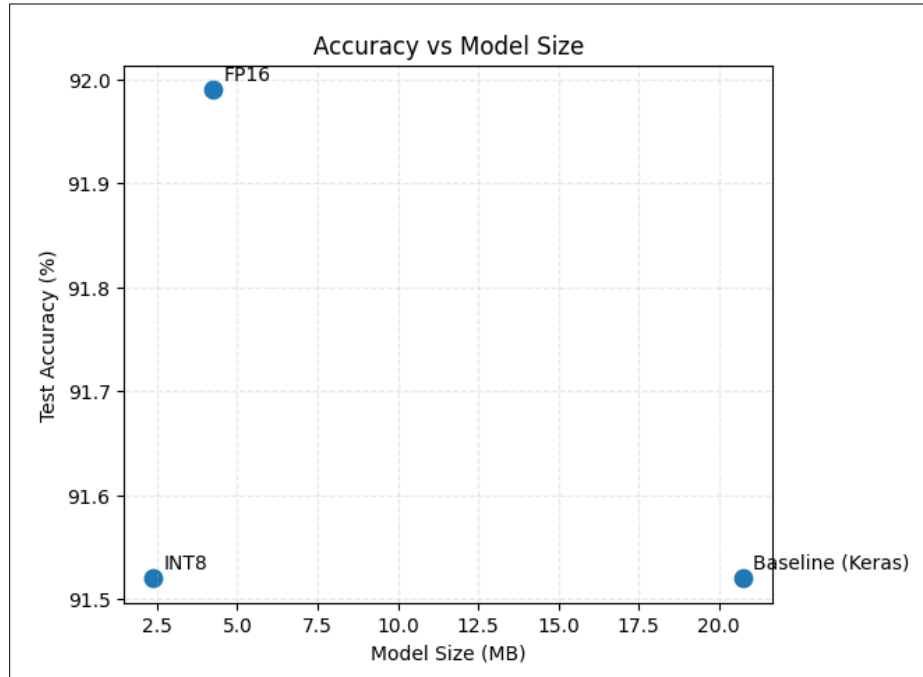


Figure 10 Relationship between model size and test accuracy for the evaluated model variants. The FP16 model preserves baseline-level accuracy while substantially reducing storage size.

As shown in Figure 10, the FP16 model provides the most favorable balance between compactness and accuracy, making it the most suitable choice for mobile deployment.

4.5. On-Device Latency

Table 9 reports mean inference latency for the FP16 TFLite model on four Android platforms. The Samsung Galaxy A55 achieves the fastest time (128.5 ms), while the Redmi Note 11 is slower (316.5 ms). All devices run well under 500 ms, suitable for real-time screening.

Table 9 Mean inference latency across tested devices

Device	Architecture	Mode	Mean ms/sample
Samsung Galaxy A55	ARM64 (Exynos 1480)	FP16	128.5 ms
Redmi Note 11	ARM64 (Snapdragon 680)	FP16	316.5 ms
Infinix	ARM64 (MediaTek Helio G96)	FP16	276.5 ms
Android Emulator (x86)	CPU	FP16	92.7 ms

Figure 11 Screenshots of the mobile application showing image selection, classification output, and probability distribution.

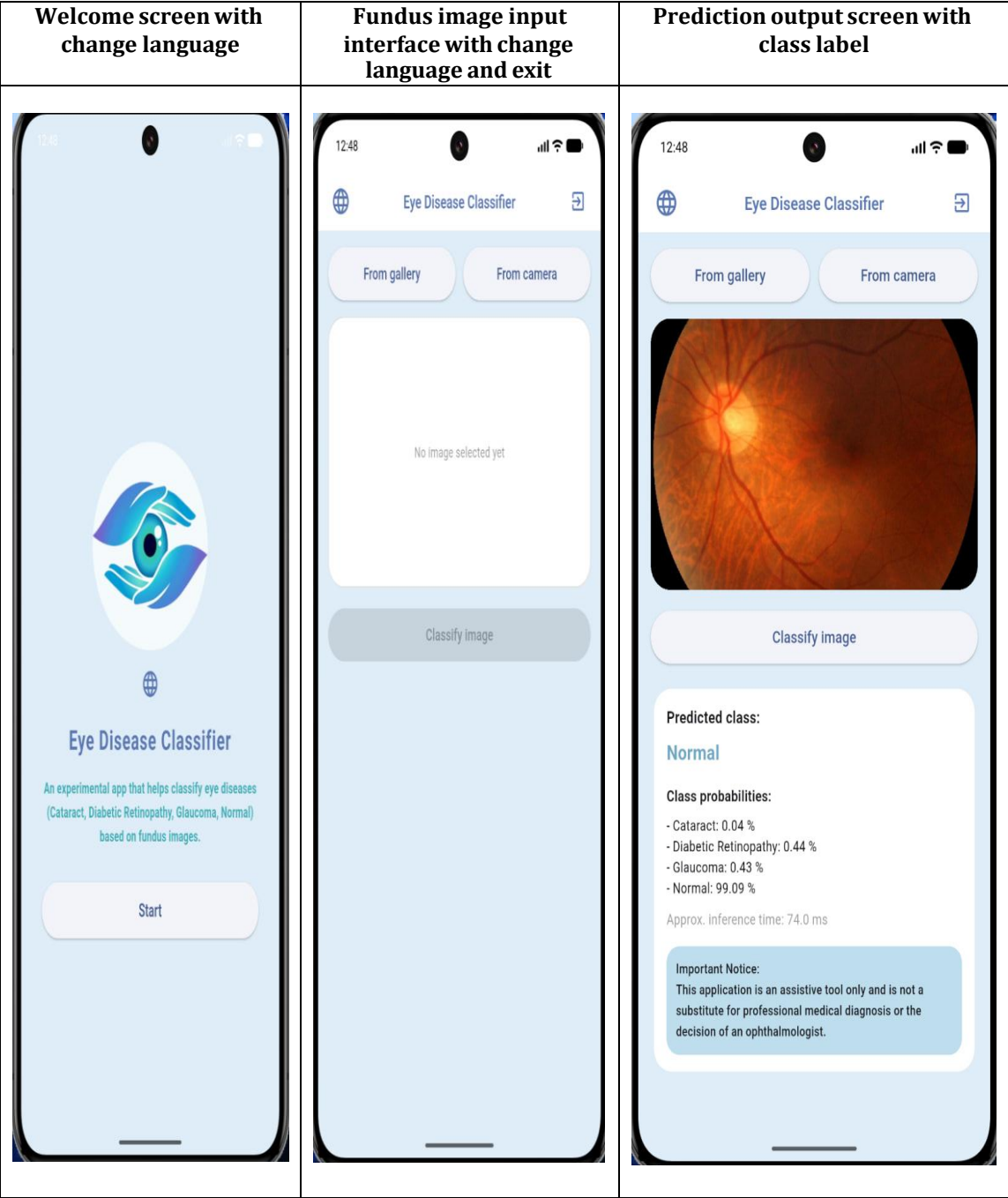


Figure 11 Screenshots from the deployed Flutter mobile application

4.6. Consistency Between Environments

We verified that the same test image produces identical predictions (within floating-point tolerance) in Colab and on-device, confirming that the preprocessing pipeline and FP16 inference are faithful to the original model.

To validate deployment correctness, inference was performed on Google Colab execution environment and Android mobile application, an example of this is as shown in Figure 12:

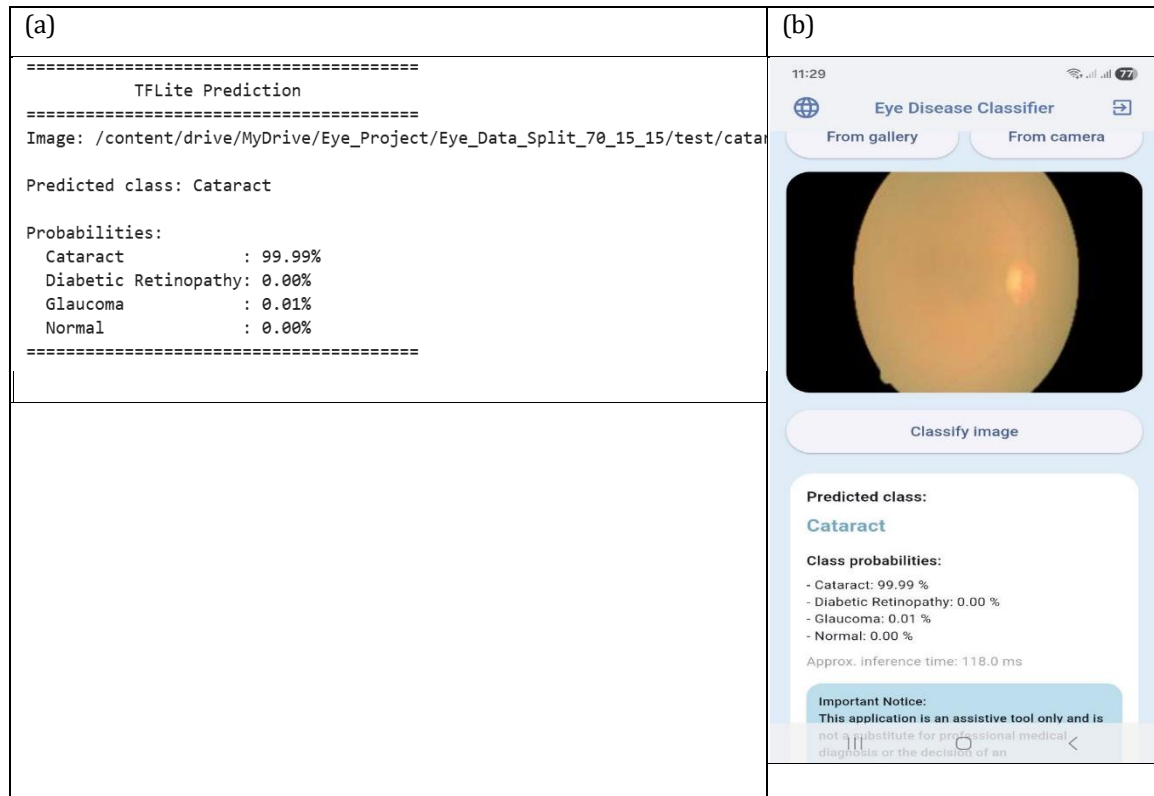


Figure 12 Comparison of prediction for the same image between the Google Colab environment (a) and the Android mobile application (b)

5. Discussion

The results demonstrate that MobileNetV2_Eye, a lightweight CNN with only 2.26M parameters, achieves competitive accuracy (91.99% after FP16 quantization) while being suitable for on-device deployment. The comparison with EfficientNetB0 shows that a simpler architecture can generalize better on this dataset, possibly due to lower overfitting.

Among optimization techniques, FP16 quantization stands out as the most stable and effective. It reduces model size by 5× and latency by an order of magnitude with no accuracy loss—in fact, a slight increase. Dynamic-range INT8 offers further compression but at the cost of accuracy (or no gain), while full-integer INT8 causes noticeable degradation. Pruning and knowledge distillation were also evaluated but did not outperform FP16 in the accuracy–size trade-off. This underscores that for medical applications, accuracy-preserving optimization is more critical than aggressive compression. On-device latencies (128–316 ms) are acceptable for interactive screening. The Samsung A55's superior performance reflects its modern Exynos chip; even mid-range devices like Redmi Note 11 run under 0.5 seconds. This confirms that the model can operate in real-world conditions without specialized hardware.

The persistent challenge with glaucoma classification ($F1 \approx 0.83$) suggests that fundus images alone may not capture all necessary cues. Future work could integrate additional modalities (e.g., OCT) or attention mechanisms focused on the optic disc.

Compared to prior works achieving 95–98% accuracy with heavy models, our approach sacrifices a few percentage points for massive gains in deployability. In resource-constrained settings, this trade-off is pragmatic: a tool that works offline on existing smartphones can reach more patients than a cloud-based system requiring high-end infrastructure.

The combination of lightweight architecture and stable FP16 quantization demonstrates that clinically relevant screening tools can be implemented directly on commodity smartphones. This is particularly important for rural screening programs where internet connectivity and specialized hardware are unavailable.

The proposed system should be considered an assistive screening tool rather than a standalone diagnostic system. Final clinical decisions should remain under the supervision of qualified ophthalmologists.

These findings highlight an important practical trade-off in mobile medical AI: the most useful model is not necessarily the one with the highest raw complexity, but the one that offers the best balance between predictive reliability, compact storage, and real-world deployability. In this study, the FP16 variant achieved that balance most effectively, showing that lightweight optimization can preserve strong classification performance while making offline smartphone deployment feasible.

6. Limitations

Despite the promising results, several limitations should be acknowledged.

First, the dataset used in this study contains 4,217 images, which is relatively small compared with large-scale clinical retinal datasets such as EyePACS.

Second, the evaluation was performed using a single stratified train-validation-test split rather than cross-validation, and no external validation dataset was used. As a result, the reported performance may not fully reflect generalization across different populations or imaging conditions.

Third, the model relies solely on color fundus images. Certain diseases such as glaucoma may benefit from additional modalities such as optical coherence tomography (OCT) or visual field testing. Finally, although real-device latency measurements were conducted, the proposed system has not yet been evaluated in a real clinical workflow with healthcare professionals.

7. Conclusion and Future Work

We have presented a complete pipeline from lightweight model selection to on-device deployment for eye disease classification. MobileNetV2_Eye achieves 91.99% accuracy after FP16 TensorFlow Lite conversion, with a 4.26 MB model and 128–316 ms inference latency on mid-range Android phones. Systematic comparison with EfficientNetB0 and multiple optimization techniques confirms that FP16 quantization offers the best balance of accuracy, size, and speed. The Flutter-based mobile application runs fully offline, making it suitable for screening in low-resource environments.

Future work will focus on:

- Improving glaucoma detection via attention mechanisms or multimodal data.
- Integrating uncertainty estimation for safer clinical decision support.
- Expanding the dataset with more diverse fundus images and external validation.

Future work will evaluate the proposed system on larger and more diverse datasets and explore external validation using datasets such as EyePACS or ODIR to further assess generalization.

Declarations

- **Ethics approval and consent to participate:** Not applicable. This study used a publicly available dataset of anonymized fundus images obtained from Kaggle, and did not involve any interaction with human participants or access to identifiable private information.
- **Consent for publication:** Not applicable.

- **Availability of data and materials:** The dataset used in this study is publicly available on Kaggle: Eye Diseases Classification (<https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification>). The trained model files and code are available from the corresponding author upon reasonable request.
- **Competing interests:** The authors declare that they have no competing interests.
- **Funding:** The authors received no specific funding for this work.
- **Authors' contributions:** M.F. Conceptualized the study, designed the methodology, implemented the software and training pipelines, conducted the formal analysis and validation, curated the data, prepared the visualizations, and wrote the original draft. All work was conducted under the supervision and guidance of Dr. Bassel Mustaffa AlKhatib. All authors reviewed and edited the final manuscript. All authors read and approved the final manuscript.
- **Acknowledgements:** The author would like to express sincere appreciation to Dr. Bassel Mustaffa AlKhatib for his valuable supervision, constructive feedback, and continuous academic support throughout all stages of this research. His guidance greatly contributed to the development and completion of this work.
- **Clinical Trial Number:** Not applicable.

References

- [1] G. Brian and H. Taylor, "Cataract blindness—challenges for the 21st century," *Bulletin of the World Health Organization*, pp. 249-256, 2001.
- [2] L. S. Lim, P. Mitchell, J. M. Seddon, F. G. Holz and T. Y. Wong, "Age-related macular degeneration," *The Lancet*, pp. 1728-1738, 2012.
- [3] R. N. Weinreb, T. Aung and F. A. Medeiros, "The pathophysiology and treatment of glaucoma: a review," *JAMA*, pp. 1901-1911, 2014.
- [4] M. D. Abràmoff, M. K. Garvin and M. Sonka, "Retinal imaging and image analysis," *IEEE Reviews in Biomedical Engineering*, pp. 169-208, 2010.
- [5] World Health Organization, "World report on vision," Geneva: WHO, 2023.
- [6] D. S. Kermany and et al, "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning," *Cell*, pp. 1122-1131, 2018.
- [7] V. Gulshan and et al, "Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs," *JAMA*, pp. 2402-2410, 2016.
- [8] D. S. W. Ting and et al, "Development and Validation of a Deep Learning System for Diabetic Retinopathy and Related Eye Diseases Using Retinal Images from Multiethnic Populations With Diabetes," *JAMA*, pp. 2211-2223, 2017.
- [9] J. De Fauw and et al, "Clinically applicable deep learning for diagnosis and referral in retinal disease," *Nature Medicine*, pp. 1342-1350, 2018.
- [10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, 2018.
- [11] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, USA, 2019.
- [12] G. V. Doddi, "Eye Diseases Classification," Kaggle, 2022. [Online]. Available: <https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification>.

- [13] A. G. Howard and et al, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017.
- [14] X. Zhang, X. Zhou, M. Lin and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 2018.
- [15] A. K. B. Arnob, M. H. R. Chayon, F. Al Farid, M. N. Husen and F. Ahmed, "A Lightweight CNN for Multiclass Retinal Disease Screening with Explainable AI," J. Imaging, pp. 45-58, 2025.
- [16] H. Wang and et al, "Many-MobileNet: Multi-Model Augmentation for Robust Retinal Disease Classification," arxiv:2412.02825 (cs), 2024.
- [17] A. Zafar, K. S. Kim, M. U. Ali, J. H. Byun and S.-H. Kim, "A lightweight multi-deep learning framework for accurate diabetic retinopathy detection and multi-level severity identification," Frontiers in Medicine, p. 112345, 2025.
- [18] S. M. Saqib I. M. Qureshi and M. A. Khan, "Cataract and glaucoma detection based on Transfer Learning using MobileNet," Heliyon, pp. 2405-8440, 2024.
- [19] H. Fu and et al, "Disc-Aware Ensemble Network for Glaucoma Screening from Fundus Image," IEEE Transactions on Medical Imaging, pp. 2493-2501, 2018.
- [20] J. Sahlsten and et al, "Deep Learning Fundus Image Analysis for Diabetic Retinopathy and Macular Edema Grading," Scientific Reports, p. 10750, 2019.
- [21] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems (NeurIPS), Lake Tahoe, Nevada, USA, 2012.
- [22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016.
- [23] G. Huang, Z. Liu, L. v. d. Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017.
- [24] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016.
- [25] J. Deng and et al, "ImageNet: A Large-Scale Hierarchical Image Database," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, Florida, USA, 2009.
- [26] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering, pp. 1345-1359, 2010.
- [27] C. Szegedy and et al, "Going deeper with convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015.

ABOUT AUTHORS

Manya Jamil Farah is an Information Technology graduate, and a student in the Master's program in Web Science at the Syrian Virtual University.

Email Address: manya_324407@svuonline.org

Dr. Bassel Mustaffa Alkhatib holds a PhD in Computer Engineering from Purdue University (France) and has extensive experience in artificial intelligence, information systems, and higher education. He has served as Head of the Artificial Intelligence Department at the University of Damascus and Director of the Web Sciences Master's Program at the Syrian Virtual University.

Email Address: t_balkhatib@svuonline.org