Hardware-Efficient Pruned CNN Optimized by Neural Architecture Search and Genetic Algorithm for Diabetic Retinopathy Detection on STM32F746

Omid Askari Haddad
Department of Computer Engineering
Ferdowsi University of Mashhad
https://orcid.org/0009-0006-4122-0079

Sara Ershadi Nasab
Corresponding author
Ferdowsi University of Mashhad
https://orcid.org/0000-0003-1561-4446

Scalable screening for diabetic retinopathy remains difficult to deliver where it is most needed. We present a hardware-efficient convolutional neural network (CNN) designed for reliable, on-device DR detection. Our approach integrates a compact CNN architecture with structured pruning and a Non-dominated Sorting Genetic Algorithm II (NSGA-II) to work under strict memory and compute budgets. We validate the approach on standard DR datasets and demonstrate deployment on an ARM microcontroller, highlighting its practical feasibility for portable screening tools in rural and underserved clinics. This work contributes (1) an end-to-end, resource-aware pipeline that couples architecture search with pruning, (2) a principled optimization strategy that balances diagnostic accuracy and efficiency, and (3) an embedded deployment that illustrates scalable, accessible AI-driven DR screening.

Keywords— Diabetic retinopathy detection, Genetic algorithm, Convolutional neural network, NSGA-II algorithm, ARM microcontroller

I. INTRODUCTION

A. Background

Diabetic retinopathy (DR) remains a major barrier to vision health. The International Diabetes Federation (IDF) estimates that 589 million adults aged 20–79 were living with diabetes in 2024; this number may reach 853 million by 2050. DR, a microvascular complication caused by retinal vessel damage, affects about 23% of people with diabetes worldwide. Approximately 6% have proliferative DR, 5% have diabetic macular edema, and 11% face sight-threatening disease. Despite progress in screening, access remains limited in many low- and middle-income countries due to cost and shortages of trained personnel [1].

B. Limitations of Current Screening

Traditional screening relies on clinical fundus examination by ophthalmologists or trained healthcare professionals using direct ophthalmoscopy or slit-lamp biomicroscopy [2]. Fundus photography, particularly the seven-field Early Treatment Diabetic Retinopathy Study (ETDRS) protocol, offers high sensitivity and specificity for clinically significant DR. However, these methods require specialized equipment and training, which restricts availability in rural or underserved settings. Diagnostic performance also depends on grader expertise, making consistency across sites difficult to ensure.

C. Opportunity for AI and Edge Devices

Recent advances in machine learning (ML) and deep learning (DL) help address these gaps. CNNs show strong performance in medical image analysis, including ophthalmic disease detection. AI-based DR screening can triage cases and flag patients who need specialist review. A notable example is IDx-DR, the first FDA-approved autonomous DR diagnostic system for retinal photographs, which can detect DR without specialist input. These systems are designed to support, not replace, clinicians, but they signal an important shift in DR detection and prevention [3].

Deploying high-performing CNNs on edge devices remains challenging. Models such as ResNet, Inception, and EfficientNet often contain tens of millions of parameters and require billions of operations. Such demands exceed the memory and compute budgets of low-cost hardware found in mobile phones, portable fundus cameras, and microcontrollers. Power consumption and latency also matter in the field. Cloud inference is not always feasible due to unreliable connectivity or privacy concerns. As a result, on-device processing is crucial, and there is a clear need for lightweight CNNs that preserve accuracy within tight resource limits.

Model compression can help. Pruning removes relatively unimportant filters, neurons, or connections to cut memory use and accelerate inference. Yet pruning must be controlled. Excessive pruning harms accuracy; insufficient pruning leaves efficiency gains unrealized. Finding the right accuracy–efficiency balance is therefore a key problem for DR detection on constrained hardware. Hyperparameter tuning further complicates this task because the search space is large and interdependent, making manual or grid search inefficient.

Genetic algorithms (GAs) are well suited to this type of optimization. They perform guided stochastic search by encoding model parameters as chromosomes and evolving candidate solutions through selection, crossover, and mutation [4]. GAs have been used to optimize CNN hyperparameters, design architectures, and drive structured filter pruning. With an appropriate fitness function, they can push models toward better trade-offs between accuracy and efficiency.

D. Contributions of This Work

In this work, we propose an integrated framework for accurate and efficient DR detection on resource-constrained edge devices. Our method combines:

1. a compact CNN architecture tailored for DR detection.
2. iterative pruning to remove redundant parameters while preserving essential features.
3. GA-based optimization to fine-tune hyperparameters and pruned architectures for maximum performance under strict hardware constraints.

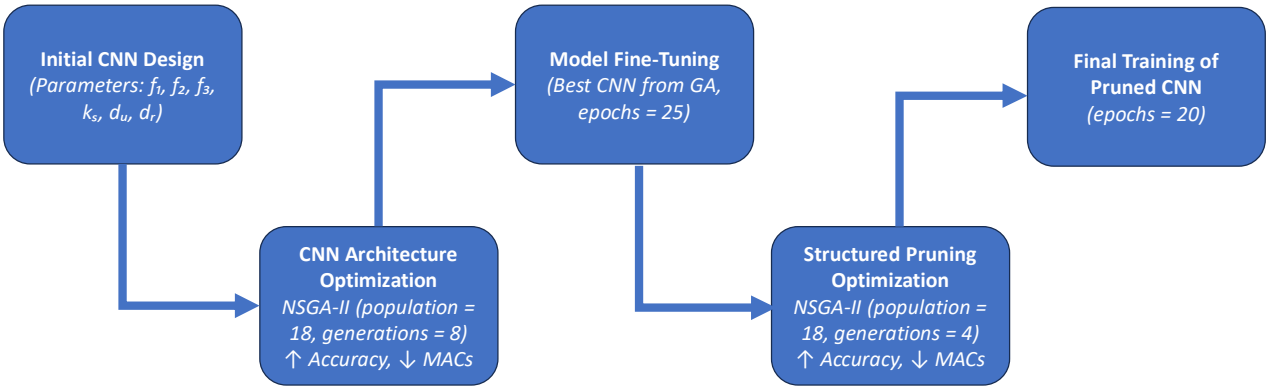


Figure 1: The framework of the proposed method.

We deploy the optimized CNN on an ARM-based microcontroller and demonstrate feasibility for real-time binary DR detection in low-resource environments. The resulting model has a small memory footprint and low computational cost, enabling portable screening tools that operate without cloud connectivity and addressing both infrastructure and privacy barriers.

The remainder of this paper is organized as follows: Section II presents related works, Section III describes the methodology, Section IV reports experimental results, Section V compares our approach with existing models, Section VI details deployment on ARM microcontrollers, and Section VII concludes the paper.

II. Related Work

A. Early CNN-Based Approaches for DR Detection

Early deep learning studies showed that CNNs can automate DR screening. After the 2015 Kaggle DR competition, several groups validated CNNs on fundus images. Manoj and Bhosale [5] trained a CNN and reported 95% sensitivity and 75% accuracy on 5,000 validation images. Gulshan et al. [6] built a deep CNN with ~90% sensitivity and 98% specificity for referable DR, reaching ophthalmologist-level performance. Lam et al. [7] used a GoogLeNet-based classifier and achieved 95% sensitivity and 96% specificity on a binary DR task.

B. Advances in DR Detection Models

Recent work has further improved accuracy. Alyoubi et al. [8] combined lesion localization and classification. Their custom CNN (CNN512) with a YOLOv3 detector achieved ~89% accuracy, 89% sensitivity, and 97.3% specificity. Çınarer et al. [9] fine-tuned DenseNet201 and ResNet152 on APTOS 2019 and reported an area under the receiver-operating-characteristic curve (AUC) up to 0.94 and ~82.7% accuracy. Moustari et al. [10] proposed an attention-guided dual-branch CNN that reached 98.5% accuracy (AUC 0.998) on 5-class grading, surpassing a DenseNet-121 baseline at 97.5% accuracy. Many recent studies report 94–96% accuracy on benchmarks by pairing advanced backbones (e.g., EfficientNet, Inception-ResNet) with data augmentation and class balancing. Akhtar et al. [11] introduced RSG-Net with extensive preprocessing and augmentation on Messidor and claimed >99% test accuracy for 4-class grading. Such extreme results may reflect dataset constraints, but the trend is clear: attention mechanisms, ensembles, and stronger training pipelines are pushing toward reliable screening.

C. Lightweight CNN Models for Efficient DR Screening

As accuracy matured, attention shifted to efficiency. Das and Pumrin [12] evaluated MobileNet and MobileNetV2 for DR classification and showed that compact models can perform well with further tuning. Zafar et al. [13] built a two-stage lightweight framework for DR severity identification and adjusted network depth to balance speed and accuracy. Akhtar et al. [11] used SqueezeNet in a blockchain-based DR system and obtained ~94% accuracy with a small model size. Fu et al. [14] designed frequency-recalibrated lightweight networks for lesion segmentation to speed up inference. Qasim et al. [15] proposed a ShuffleNetV2-based classifier for mobile use, cutting parameters by ~28% compared with MobileNetV2 and reducing inference time from 73 ms to 40 ms per image.

D. Evolutionary Algorithms in DR CNN Optimization

Evolutionary methods have also improved DR CAD systems. Researchers use GAs for feature selection and CNN hyperparameter tuning. Welikala et al. [16] applied a GA to select discriminative fundus features and paired them with an ensemble classifier to enhance proliferative DR detection. Mounika and Ravisankar [17] built a hybrid VGG16–EfficientNet model with channel attention and used a GA to optimize image preprocessing; the system reached 95% accuracy for 2-class DR. GAs can search large configuration spaces without exhaustive manual effort. By encoding parameters or architecture choices as chromosomes, they evolve models via selection, crossover, and mutation [4]. In DR detection, this approach has produced CNNs that outperform hand-crafted baselines on multiple metrics. For example, Das and Saha [18] used a GA to automatically set optimal CNN hyperparameters.

III. Methodology

We propose a resource-aware, end-to-end pipeline for binary DR detection that combines data preprocessing, automated neural architecture search, and structured model compression. The workflow is shown in Algorithm 1 and in Fig. 1.

A. Dataset

- Training: APTOS 2019 Blindness Detection [19], binarized labels.
- External validation: IDRiD [24], same binarization to test generalization on a distinct dataset.

Algorithm 1: Proposed pipeline.

Algorithm 1: Multi-Objective CNN Architecture and Pruning Optimization

Input: Training data (X_{train}, Y_{train}), validation data (X_{val}, Y_{val}), number of classes n_{cls}, input shape s, population size μ, offspring size λ, CNN generations G_{cnn}, pruning generations G_{prune}

Output: Optimized and pruned CNN model

Phase 1: CNN Architecture Search
Initialize toolbox `tb_cnn` with gene parameters: $\{f_1, f_2, f_3, k_s, d_u, d_r\}$;
Register fitness function: accuracy vs. MACs using partial training and evaluation ;
Run NSGA-II for G_{cnn} generations with population μ and offspring λ ;
Obtain Pareto front $\mathcal{P}_{cnn}$; select best individual g^*_{cnn} maximizing accuracy and minimizing MACs ;
Phase 2: Fine-Tune Base CNN
Construct CNN using g^*_{cnn} ;
Train for 25 epochs with full training data and save weights ;
Phase 3: Pruning Parameter Search
Initialize toolbox `tb_prune` with gene parameters: $\{i_{sp}, f_{sp}, b_s, e_s\}$;
Register fitness function using accuracy and MACs after pruning training ;
Run NSGA-II for G_{prune} generations ;
Obtain Pareto front $\mathcal{P}_{prune}$; select best pruning policy g^*_{prune} ;
Phase 4: Final Training with Pruning
Apply dynamic sparsity pruning schedule using g^*_{prune} ;
Train the CNN for 20 epochs with pruning applied ;
return final model and optimal gene sets $\{g^*_{cnn}, g^*_{prune}\}$

B. Preprocessing

For each input image:

1. load raw PNG, convert to RGB, and resize.
2. apply CLAHE to enhance local contrast [20].
3. use a 5×5 Gaussian blur to attenuate sensor noise.
4. apply a sharpening kernel to restore edges.
5. normalize pixel values to [0,1].

C. Architecture Design

The base architecture is deliberately compact: two depthwise-separable convolutional blocks feed into global average pooling, followed by a single dense hidden layer and a softmax head. We define a six-parameter genome comprising three convolutional filter widths, the kernel size, Number of units in the dense layer and dropout rate. We run NSGA-II with a population of 18 for 8 generations. The multi-objective fitness jointly maximizes validation accuracy and minimizes theoretical multiply–accumulate operations (MACs).

To control evaluation cost, each candidate trains for four epochs on a 30% data subsample. Candidates exceeding 0.80 validation accuracy are then trained for four additional epochs on the full dataset. We retain the Pareto frontier and select the model with the best accuracy–cost trade-off. That model is fine-tuned for 25 epochs to obtain the base network.

D. Pruning Strategy

Starting from the fine-tuned base network, we search pruning schedules that progressively increase sparsity during training while targeting low MACs. A second NSGA-II run (population 18, generations 4) explores candidate schedules; a cubic interpolation function governs the evolution of sparsity from its initial to final value. We adopt a monotonic, mask-based structured pruning rule: global sparsity increases from 0 up to 0.90, and once a weight is zeroed it remains zero. Candidates are scored on the joint objective of accuracy and MAC reduction. The best schedule is applied and the pruned model is fine-tuned for 20 epochs, yielding a highly sparse yet accurate network.

E. Optimization

For the architecture optimization stage, a population size of 18 over 8 generations was adopted after pilot experiments showed that this configuration consistently produced stable Pareto fronts within the available compute budget. Increasing either parameter beyond this point led to minimal improvements in accuracy–efficiency trade-offs while significantly increasing runtime, indicating diminishing returns. Conversely, smaller budgets failed to adequately cover the search space, resulting in less diverse and less competitive solutions.

In the pruning schedule optimization stage, a population size of 18 and 4 generations proved sufficient to converge to effective sparsity schedules. The search space in this stage was smaller and smoother, allowing good solutions to emerge quickly. Trials with larger budgets provided negligible accuracy gains relative to the added cost, while smaller budgets reduced stability across runs.

These budget choices reflect a deliberate trade-off: providing enough evaluations to ensure consistent convergence and solution diversity, while keeping the total runtime practical for iterative experimentation and aligned with the resource-aware nature of the proposed system.

F. Training Setup and Reproducibility

This section outlines the key setup used for model training, including software, hardware, and optimization parameters.

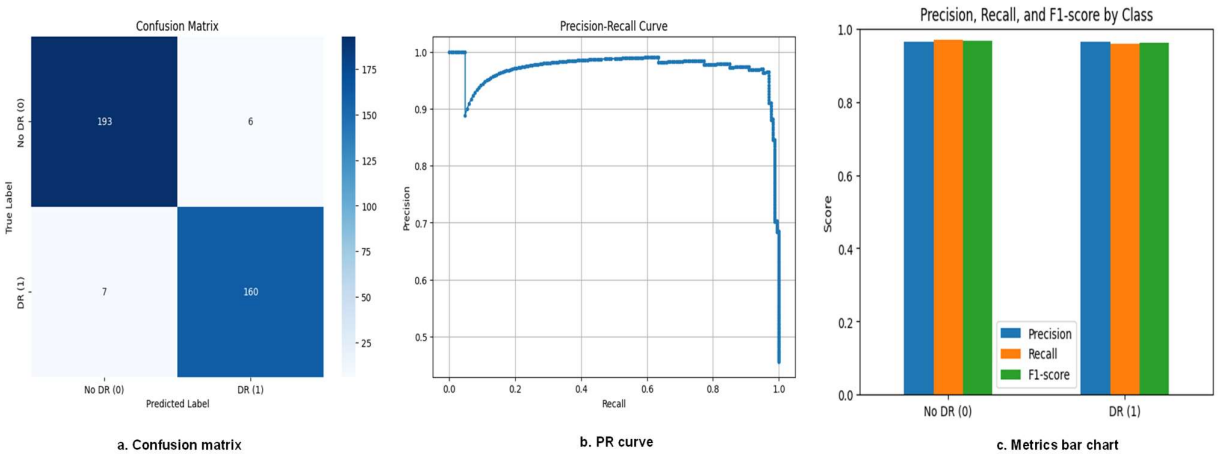
- Framework: TensorFlow/Keras 2.18 (Python 3.11)
- Hardware: We ran all training on a single NVIDIA Tesla P100-PCIE-16GB GPU (provided by Kaggle notebook).
- Optimizer: Adam.
- Loss: categorical cross-entropy.
- Learning-rate schedule: 5×10^{-4} for base training (25 epochs) and pruning (20 epochs).

We repeat the full pipeline five times with different seeds and report mean, standard deviation, and 95% confidence intervals (using Student's t-distribution).

IV. Results

A. CNN architecture parameters

In the initial phase of the optimization process, the CNN hyperparameters (filter sizes, kernel size, dense layer units, and dropout rate) were tuned using the NSGA-II algorithm. The optimal set of parameters obtained from this search is summarized in Table I.



a. Confusion matrix b. PR curve c. Metrics bar chart

Figure 2: Performance Visualizations of the Optimized CNN Model

TABLE I: Best Hyperparameters for CNN

Hyperparameter	Meaning	Optimum Value
F1	Number of filters in first layer	20
F2	Number of filters in second layer	16
F3	Number of filters in third layer	249
KS	Kernel size	3
DU	Number of units in the dense layer	73
DR	Dropout rate	0.45

B. Pruning parameters

The best pruning parameters from the Pareto front, obtained via NSGA-II optimization, are presented below. These parameters were selected to achieve an optimal trade-off between accuracy and computational efficiency in the lightweight CNN for DR classification, enabling high validation accuracy while maintaining low resource requirements.

- Initial Sparsity (isp): 0.00
- Final Sparsity (fsp): 0.90
- Begin Step (bs): 184
- End Step (es): 1564

C. Final model performance

The optimized model achieved high test and validation accuracy with minimal validation loss. Due to its low parameter count, the model exhibits a compact memory footprint and reduced computational complexity making it suitable for deployment on resource-constrained platforms such as mobile devices or embedded systems. The detailed performance and efficiency metrics are reported in Table II. Additionally, the model's performance is visualized using a precision–recall curve, a confusion matrix, and bar plots summarizing precision, recall, and F1-score, as shown in Fig. 2.

V. Comparison

To place the proposed model in context with existing DR detection systems, its performance is compared against four widely used baseline architectures: VGG-16, MobileNetV2, ResNet-50, and the recently published framework by Zafar et al. [13]. The proposed architecture requires only tens of thousands of parameters, representing an order-of-magnitude reduction relative to conventional backbone networks. This substantial model compression is achieved without degrading discriminative capability; despite its ultra-compact footprint, the model maintains classification performance comparable to that of substantially larger networks. The complete comparative results are reported in Table III.

TABLE II: Performance and Efficiency Metrics

Metric	Mean ± Sd	95% CI
Test Accuracy	94.0 ± 2.1	92.8–95.2
Validation Accuracy	94.5 ± 0.7	94.1–94.9
Validation Loss	0.1355	
Total Parameters	25,027	
Model Size (Kilobytes)	150.26	
MACs	1,633,670	

TABLE III: Binary DR detection on APTOS 2019

Network	Accuracy (%)	Parameters (M)
VGG-16 [21]	79.99	138
MobileNetV2 [22]	97	3.5
InceptionResNetV2[25]	97.54	54
Xception [25]	97.81	20
RA-EfficientNet [25]	98.36	4.27
Zafar et al.[13]	99.06	1.3
Proposed	96.1	0.025

VI. Deployment on ARM Microcontroller

Deploying AI models on microcontrollers poses significant challenges due to limited resources. For example, the STM32F746G Discovery board uses an ARM Cortex-M7 microcontroller. It has only 1,024 KB of flash memory and 320 KB of RAM. Conventional AI workloads often exceed these limits. They require more memory and processing power. As a result, real-time inference becomes difficult without external support like cloud connectivity. Our optimized model overcomes these constraints. We verified

this with X-CUBE-AI analysis. This efficiency allows real-time DR detection on the device. It suits resource-limited settings, such as rural clinics [23]. The deployment process was straightforward. We converted the trained .h5 model to C code using X-CUBE-AI v10.1. Then, we integrated it into STM32CubeIDE v1.19.0 for firmware development. For testing, we set up a PC terminal. It communicated via UART to send retinal fundus images to the board. The board returned classification outputs. Results appeared on the terminal and the board's LCD. This setup provided user-friendly interaction. Table IV summarizes the key resource usage from X-CUBE-AI analysis.

TABLE IV: Key resource usage from X-CUBE-AI analysis.

Resource	Component	Usage (Bytes)
Flash	Weights	34,312
Flash	Runtime Library	19,422
Flash	Total	53,734
RAM	Activations	271,424
RAM	Runtime Library	5,696
RAM	Total	277,120

These metrics show the model's feasibility for edge inference. This enables affordable, portable DR screening. The solutions work without cloud dependency. See Figs. 3 and 4 for analysis results and performance.

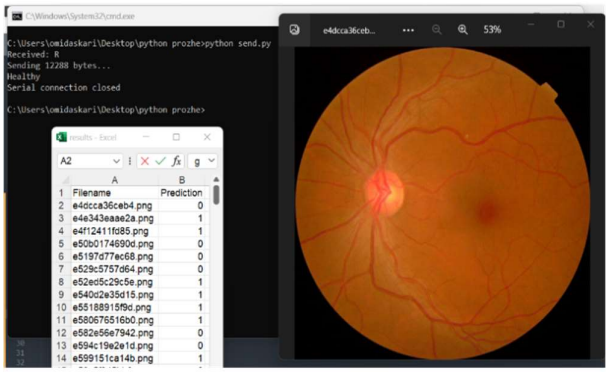


Figure 3: Result of the model's correct prediction on UART: the fundus image is healthy and is correctly classified

Figure 4: Result of the model's prediction on LCD, which occurs simultaneously with UART.

VII. Conclusion

This work shows that co-designing neural architectures and compression schedules with the target hardware can deliver clinically strong, on-device diabetic retinopathy screening under tight resource limits. We introduce an end-to-end, resource-aware pipeline that jointly optimizes a compact CNN and its structured-pruning schedule via NSGA-II with explicit budgets for accuracy, MACs, and memory. A monotonic, mask-based pruning strategy maintains sparsity throughout training to produce an ultra-compact yet discriminative model. The system is fully embedded on a low-cost microcontroller supporting real-time inference. Empirically, it is robust across runs, achieving mean test accuracy of more than 94%, while cutting parameters by orders of magnitude versus conventional backbones. External validation and deployment metrics indicate feasibility without cloud or specialized accelerators, enabling scalable, privacy-preserving triage in low-resource settings.

Looking ahead, this foundation opens several promising avenues for extension: (1) Advancing from binary to multi-class severity grading. (2) Integrating quantization to further minimize memory demands. (3) We should try and conduct real-world tests with handheld fundus cameras to check workflow fit, cost-effectiveness, and whether performance holds across varied imaging conditions.

References

[1] International Diabetes Federation, *IDF Diabetes Atlas*, 11th ed. Brussels, Belgium: IDF, 2025. [Online]. Available: https://diabetesatlas.org/media/uploads/sites/3/2025/04/IDF_Atlas_11th_Edition_2025-1.pdf. Accessed: Sep. 28, 2025. Diabetes Atlas

[2] J. W. Y. Yau *et al.*, "Global prevalence and major risk factors of diabetic retinopathy," *Diabetes Care*, vol. 35, no. 3, pp. 556–564, Mar. 2012, doi: 10.2337/dc11-1909. Europe PMC

[3] M. Savoy, "IDx-DR for diabetic retinopathy screening," *Am. Fam. Physician*, vol. 101, no. 5, pp. 307–308, Mar. 2020. American Academy of Family Physicians+1

[4] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992. [Online].Available:https://mitpress.mit.edu/9780262581110/adaptation-in-natural-and-artificial-systems/. Accessed: Sep. 28, 2025. MIT Press

[5] S. H. Manoj and A. A. Bhosale, "Detection and classification of diabetic retinopathy using deep learning algorithms for segmentation to facilitate referral recommendation for test and treatment prediction," *arXiv:2401.02759*, 2024. [Online]. Available: https://arxiv.org/abs/2401.02759. Accessed: Sep. 28, 2025.

[6] V. Gulshan *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *JAMA*, vol. 316, no. 22, pp. 2402–2410, Dec. 2016, doi: 10.1001/jama.2016.17216. JAMA Network

[7] C. Lam, D. Yi, M. Guo, and T. Lindsey, "Automated detection of diabetic retinopathy using deep learning," *AMIA Jt. Summits Transl. Sci. Proc.*, vol. 2017, pp. 147–155, 2018.

[8] W. L. Alyoubi, M. F. Abulkhair, and W. M. Shalash, "Diabetic retinopathy fundus image classification and lesions localization system using deep learning," *Sensors*, vol. 21, no. 11, Art. no. 3704, May 2021, doi: 10.3390/s21113704. MDPI

[9] G. Çınarer, K. Kılıç, and T. Parlar, "A deep transfer learning

framework for the staging of diabetic retinopathy," *J. Sci. Reports-A*, no. 51, pp. 106–119, Dec. 2022. [Online]. Available: https://dergipark.org.tr/en/download/article-file/2615453. Accessed: Sep. 28, 2025. DergiPark

[10] A. M. Moustari, Y. Brik, B. Attallah, and R. Bouaouina, "Two-stage deep learning classification for diabetic retinopathy using gradient weighted class activation mapping," *Automatika*, vol. 65, no. 3, pp. 1284–1299, 2024, doi: 10.1080/00051144.2024.2363692. MDPI

[11] S. Akhtar *et al*., "A deep learning based model for diabetic retinopathy grading," *Sci. Rep.*, vol. 15, Art. no. 3763, 2025, doi: 10.1038/s41598-025-87171-9. arXiv

[12] P. K. Das and S. Pumrin, "Diabetic retinopathy classification: Performance evaluation of pre-trained lightweight CNN using imbalance dataset," *Eng. J.*, vol. 28, no. 7, pp. 13–25, 2024, doi: 10.4186/ej.2024.28.7.13. ScienceDirect

[13] 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," *Front. Med.*, vol. 12, Art. no. 1551315, Apr. 2025, doi: 10.3389/fmed.2025.1551315. etasr.com

[14] Y. Fu, M. Liu, G. Zhang, and J. Peng, "Lightweight frequency recalibration network for diabetic retinopathy multi-lesion segmentation," *Appl. Sci.*, vol. 14, no. 16, Art. no. 6941, Aug. 2024, doi: 10.3390/app14166941. SpringerLink

[15] D. K. Qasim, S. A. Jebur, L. R. Ali, A. J. M. Khalaf, and A. J. Hussain, "Lightweight convolutional neural networks for retinal disease classification," *arXiv:2506.03186*, 2025. [Online]. Available: https://arxiv.org/abs/2506.03186. Accessed: Sep. 28, 2025. actainformmed.org

[16] R. A. Welikala *et al*., "Genetic algorithm based feature selection combined with dual classification for the automated detection of proliferative diabetic retinopathy," *Comput. Med. Imaging Graph.*, vol. 43, pp. 64–77, 2015, doi: 10.1016/j.compmedimag.2015.03.003.

[17] S. Mounika and V. Ravisankar, "Diabetic retinopathy detection using the genetic algorithm and a channel attention module on hybrid VGG16 and EfficientNetB0," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 2, pp. 21319–21325, 2025, doi: 10.48084/etasr.9720.

[18] S. Das and S. K. Saha, "Diabetic retinopathy detection and classification using CNN tuned by genetic algorithm," *Multimed. Tools Appl.*, vol. 81, no. 6, pp. 8007–8020, 2022, doi: 10.1007/s11042-021-11824-w. journal.esrgroups.org

[19] N. E. M. Khalifa, M. Loey, M. H. N. Taha, and H. N. E. T. Mohamed, "Deep transfer learning models for medical diabetic retinopathy detection," *Acta Inform. Med.*, vol. 27, no. 5, pp. 327–332, Dec. 2019, doi: 10.5455/aim.2019.27.327-332.

[20] P. Musa, F. Al Rafi, and M. Lamsani, "A review: Contrast-limited adaptive histogram equalization (CLAHE) methods to help the application of face recognition," in *Proc. 2018 3rd Int. Conf. Inform. Comput. (ICIC)*, 2018, pp. 1–6, doi: 10.1109/IAC.2018.8780492. CoLab

[21] N. Yatam and P. Gera, "Diabetic retinopathy detection using VGG-16 deep learning architecture," *J. Electr. Syst.*, vol. 20, no. 7s, pp. 3620–3627, 2024. [Online]. Available: https://journal.esrgroups.org/jes/article/view/4213. Accessed: Sep. 28, 2025. journal.esrgroups.org+1

[22] A. M. Pamadi, A. Ravishankar, P. A. Nithya, G. Jahnavi, and S. Kathavate, "Diabetic retinopathy detection using MobileNetV2 architecture," in *Proc. 2022 Int. Conf. Smart Technol. Syst. Next Generation Comput. (ICSTSN)*, 2022, pp. 1–5, doi: 10.1109/ICSTSN53084.2022.9761289.

[23] STMicroelectronics, "STM32F746G-Discovery: Discovery kit with STM32F746NG MCU." [Online]. Available: https://www.st.com/en/evaluation-tools/32f746gdiscovery.html. Accessed: Sep. 28, 2025. Europe PMC

[24] P. Porwal *et al*., "Indian Diabetic Retinopathy Image Dataset (IDRiD): A database for diabetic retinopathy screening research," *Data*, vol. 3, no. 3, Art. no. 25, 2018, doi: 10.3390/data3030025. OUCI

[25] S.-L. Yi, X.-L. Yang, T.-W. Wang, F.-R. She, X. Xiong, and J.-F. He, "Diabetic retinopathy diagnosis based on RA-EfficientNet," *Appl. Sci.*, vol. 11, no. 22, Art. no. 11035, 2021, doi: 10.3390/app112211035.