A Mobile Deep Learning Classification Model for Diabetic Retinopathy

Daniel Rimaru[1], Antonio Nehme[1], Musaed Alhussein[2], Khaled Mahbub[1], Khusheed Aurangzeb[2], Anas Khan[3,*]
[1]Birmingham City University, College of Computing, Birmingham, UK
[2]Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh, 11543, Saudi Arabia
[3]Department of Electrical Engineering, University of Wah, Quaid Avenue, Wah, Rawalpindi, Punjab 47040, India
daniel.rimaru@mail.bcu.ac.uk; antonio.nehme@bcu.ac.uk; musaed@ccis.ksu.edu.sa; khaled.mahbub@bcu.ac.uk; kaurangzeb@ksu.edu.sa; anask8726@gmail.com

***Abstract*—The pupil, iris, vitreous, and retina are parts of the eye, where any defect due to physical damage or chronic diseases to these parts of the eye can lead to partial vision loss or complete blindness. Changes in retinal structure due to diabetes or high blood pressure lead to diabetic retinopathy (DR). The early diagnosis of DR using computer-aided automated tools is possible due to tremendous advancements in machine and deep learning models in the last decade. Devising and implementing innovative deep learning models for retinal structural analysis is crucial to the early diagnosis of DR and other eye diseases. In this work, we have developed a new approach, which involves the development of a lightweight convolutional neural network (CNN)-based model for segmentation of retinal vessels and a mobile application for DR grading. This paper covers the development process of an Android application that leverages the power of CNN-based deep learning model to detect DR regardless of its stage. To achieve this, two models have been created and compared, the best one having an accuracy of 96.72 %. An Android application has then been developed, that makes calls to this model and then displays the results on screen with a simple-to-understand interface developed using the Kivy framework.**

***Index Terms*—Diabetic retinopathy; Deep neural networks; Machine learning; Retinal vessel segmentation.**

I. INTRODUCTION

In recent years, medical experts have been assisted by emerging computer vision technologies. They have been using these ground-breaking tools to consolidate and validate their analysis, thus minimising human error. Massive progress has been made in the field of artificial intelligence (AI) over the last years, sometimes even outperforming experts in certain tasks. The popularity of AI comes from the ever-increasing performance in accurately providing solutions to various problems. With the emerging power of AI, people have taken massive steps towards automating the eye disease detection process by using deep learning (DL) technologies. Mobile devices have become one of the most common pieces of computing hardware, owned by the majority of the population. As such, it is possible to target a large audience using this platform. Moreover, most of them are permanently connected to a network, allowing information to be transferred with ease. Mobile phones have also been benefiting from advancements in hardware and more complex applications are becoming available for common use.

The pupil, iris, vitreous, and retina are constituent parts of the human eye, where any flaw in any of these constituent parts of the eye may lead to partial vision loss or full blindness. Different chronic diseases such as glaucoma, diabetic retinopathy (DR), cataracts, etc. progress gradually and damage specific parts of the eye such as retinal vessels, optic cup, optic disks, etc. The composition of several parts of human organs varies from individual to individual. Similarly, the damage to human organs due to chronic diseases also varies from individual to individual due to various factors. This indicates that certain individuals may suffer from a particular chronic illness for a long time without any effect on their vision. For other individuals, the severity of chronic diseases in different human organs is severe due to weakness of the organ due to inheritance problems or the severity of the chronic disease. There is a need for regular and inexpensive monitoring of human eyes, and the development of advanced and automated AI-based tools and mobile applications for the analysis of retinal fundus images for timely diagnoses of the eye diseases is an enabler for this. Additionally, such automated diagnostic tools and applications could be used for country-level screening programmes. These large-scale screening programmes for any disease are highly important for the detection and prevention of specific chronic diseases in the population, who are mostly unaware of the development of these diseases. Using advanced AI-based automated tools can help detect diseases early and ensure timely treatment. This will enable humanity to prevent such diseases from advancing to critical stages, where it is almost impossible to reverse them.

To enable affordable large-scale screening (population level) for diabetic retinopathy, this paper leverages the power of convolutional neural network (CNN) to develop a mobile

Manuscript received 27 June, 2024; accepted 30 October, 2024.
This research is funded by Researchers Supporting Project No. RSPD2024R553, King Saud University, Riyadh, Saudi Arabia.

application that allows the classification of retinal scans as healthy or with DR. To reach a level of accuracy of 96.72 % while keeping the application lightweight, the classification models were trained on thousands of scans on a personal computer before deploying the trained model in the mobile application.

We believe that the appropriate course of action for large-scale eye disease diagnosis would be to utilise CNN-based trained models on an augmented dataset. The MobileNetV3 model will be used for the early development stages as it allows flexibility in design decisions later on, due to the versatile nature of this architecture. When it comes to augmenting the dataset, we will attempt to implement FasterAugment using generative adversarial network (GAN) on a desktop computer. The improved dataset will then be used to create models in a mobile environment. This project finds its novelty in the combination of an augmented dataset with the MobileNetV3 model, in the context of DR detection, all packaged into an Android application. There appears to be no similar combination of procedures attempted in the literature.

The contributions of the work are given below:

– CNN-based models are used for segmentation of retinal images;

– A suitable classification model is used to grade the retinal scans as healthy and diseased;

– A mobile application is developed for Android-based platforms to enable the end users to visualise and see the results instantly;

– A publicly available Diabetic Retinopathy Dataset with 3662 images is used to evaluate the performance of the developed method;

– The performance of the developed application and model is validated using two Android Studio emulators and two physical devices.

The rest of the paper is divided as follows. Section II covers the related work. Section III covers the dataset and model implementation. Section IV discusses the development of the application. Section V includes the results and discussion, and Section VI covers the conclusions and future work.

II. Background and Related Work

In this section, we discuss the literature that is relevant to eye disease classification, as well as existing deep learning models running on mobile devices.

A. Eye Disease Classification

The work of Alyoubi, Shalash, and Abulkhair [1] features a list of publicly available datasets that can be used for this project. Given the fact that the datasets contain retinal fundus images, this means that the diseases that can be detected will be caused by retinal problems. As such, there is a limitation regarding the diseases that can be identified. Examples of such diseases are diabetic retinopathy (DR) or retinitis pigmentosa (RP). When it comes to detecting diabetic retinopathy, a novel method is to first extract the retinal vessels and save them in a separate binary (black-and-white) image [2]. Using this image, they can count the nonzero pixels and see if the number of vessels is higher than usual. In this case, they can infer "that abnormal vessels have started growing resulting into proliferative diabetic retinopathy" [3]. This approach allows classic machine learning (ML) techniques such as k-nearest neighbours (KNN) to be used with high accuracy (96.23 %). While research articles on this topic strongly suggest using CNNs for classification [3], there are some people who have tried some alternatives. One such experiment was carried out by Mo, Zhang, and Feng [4], where they applied the segmentation strategy to recognise diabetic macular edema, which is a complication of DR.

Instead of using CNN, the DL method used was a deep residual network, the point being that the classic CNN was going to be more difficult to train. Deep residual learning might be a suitable approach for our lightweight model, considering that architecture has historically been known to be easier to optimise [5]. Like, the authors in [6] have used a lightweight residual connection-based model called "Colonsgnet" to accurately segment the retinal vessels and locate the true vessels for an efficient diagnosis of DR. Another residual connection-based deep neural network (DNN) was applied in [7] to efficiently detect the optic disk (OD) and optic cup (OC) that gives the correct estimation of the disk-to-cup ratio, which is the key parameter in the detection of glaucoma and DR.

With the huge advancement in the field of DL, it is also possible to skip the segmentation step and automatically classify the image. The results also show an increase in accuracy (96.5 % to 99.7 %) [8]. The reason for this performance is also due to various data augmentation procedures. The dataset that was used was imbalanced, favouring diseased eyes. As such, some data augmentation techniques have been used such as flipping, rotating and resizing the images for the purpose of creating more images of healthy eyes, greatly improving the training data, and, by extension, the model. Another option would be to harness the power of a pretrained DL model and modify it to suit our needs. This approach was tested using two datasets which can be found on Kaggle. It yielded lower accuracy scores (72.33 % and 82.18 %) but skipped the segmentation step, which shows how important segmentation can be [9].

B. Data Augmentation

Modifying the data that are used can be a decisive factor in the generalisability of the CNN algorithms. To enhance the performance of the model, it is important to apply different data augmentation strategies, such as horizontal and vertical flipping, rotation from 0 to 180 or 0 to 360 degrees, rescaling factor, and normalisation as applied in [10] for efficient segmentation of retinal vessels. It is possible to automate the process of augmenting the data, having the best augmentation policies for a certain dataset identified and applied.

Cubuk, Zoph, Mane, Vasudevan, and Le [11] have proposed the AutoAugment algorithm that has enhanced the performance of many image classification tasks by improving the datasets [11]. This search algorithm tries all possible combinations of data augmentation policies to find the best one. The problem with this algorithm is clearly highlighted by [12]: it requires "thousands of GPU hours" to converge to an optimal. However, they do propose an improved version (Fast AutoAugment), which can greatly speed up the search time by skipping some of the possible combinations. This raises the risk of getting stuck in a suboptimal configuration, which is something that must be taken into account.

Another concept worth exploring is the generative adversarial network (GAN) framework. Here, we try to create two models. The first one is a generative model that is trying to create additional images given the training data, while the second one tries to predict whether a given image originated from the training data or the generative model. The objective is to maximise the probability that the predictor model makes a mistake [13]. This concept has been applied for the detection of diabetic retinopathy by Zhou, Wang, He, Cui, and Shao [14]. The evaluation of the experiment involved asking ophthalmologists to independently evaluate the synthesised images. The results show that the experts could correctly tell a real image from a synthesised image only 65 % of the time. This means that the generated images are similar to the originals, proving the validity of the method.

Lastly, there is the option of combining automatic data augmentation algorithms with GAN. This idea was pitched in [15], as the Faster AutoAugment algorithm. The results of this experiment have shown great performance in low-resource scenarios, which is something we are actively looking for.

C. Mobile Deep Learning Models

The integration of AI and embedded systems makes it possible to implement systems that run in real time without having to transfer all data to central servers. Although the low capacity of the embedded systems greatly hinders this integration, the ability to integrate them into a wide range of microcontrollers is a huge advantage. The integration of AI with embedded systems has attracted the attention of industry giants, including Google, which launched the TensorFlow Lite platform, which provides a set of tools that enable the user to convert neural network (NN) models into simplified and reduced versions.

Despite their high accuracy, DNNs are more computationally and memory-demanding than other ML algorithms. On the other hand, embedded systems have the least computing and memory resources, where the integration of DL and embedded systems faces many challenges. Training DL models is the most difficult challenge, as training DL models consist of dense parameters that form a heavy weight to achieve high accuracy. It is computationally expensive and consumes many resources, energy, memory, and time. However, embedded systems have not yet become efficient enough to train DL models due to limited resources. Thus, the performance of DL models will be significantly affected compared to high performance computing (HPC). Therefore, straightforward integration would create inefficient solutions.

So, first, the selection of appropriate DNN model is needed that must be followed by optimisation for developing embedded system-based AI solutions for eye disease diagnosis.

Looking at the mobile deep learning landscape, Zhang *et al.* present multiple possible models that can be used for mobile DL, including MobilenetV1-V3, DenseNet, and AlexNet [16]. Depending on the use case, certain models can be more appropriate; in fact, performance is variable depending on the hardware as well. Despite being categorized as an image classification model, the MobileNet model series can also be deployed as a semantic segmentation model. These models follow the NN architecture, but are designed for low-resource devices [17]. It is recommended to use the highest version available, given its tendency to improve as the versions increase [17].

Another possible architecture that can be used is called "ShuffleNet". Zhang, Zhou, Lin, and Sun [18] claimed that it would be 13 times more efficient than AlexNet in terms of complexity, yielding similar accuracy. The problem with this statement is that AlexNet was developed in 2012, and was already outclassed by various models such as Clarifia, VGG-16, GoogleNet, and ResNet [19]. Although the model evaluation benchmark was flawed to improve the model, the contribution of Nagda, Momaya, Pandey, Khanna, and Verma has shown that the model is still relevant, considering that its performance metrics have only been beaten by MobileNet, which has always been a top competitor in the market [20]. The third place in this evaluation was UNet, which is a specially designed biomedical image segmentation model [21].

The authors in [22] developed a mobile application for the detection and grading of DR using Google AI technologies, specifically TensorFlow and Google Cloud ML. The model is trained on 12062 fundus images using Google TensorFlow and Cloud ML. After high accuracy was achieved, this model was implemented into a mobile application.

Sheikh and Qidwai [23] used a lightweight MobileNetV2 model to inspect the severity of the DR. They used bio-inspired retinal filters and tuned the hyper parameters for the enhancement of retinal features and achieved high accuracy and area under the curve (AUC) score of 91.6 % and 0.9, respectively. Looking further, a both mobile and web application was developed in [24] for the screening of DR with the objective of providing screening facilities in rural areas as the application can work both online and offline. MobileNet and ResNet 50 architectures were used for the development of mobile applications and web application, respectively.

Another MobileNet-based lightweight model is presented to facilitate people living in rural areas of Nepal to detect DR [25]. An extensive dataset that contains distinctive DR images was utilised to fine-tune the MobileNet architecture for the accurate classification of retinopathy. In [26], transfer learning is applied using a lightweight model called "NasnetMobile" along with the configuration of multilayer perceptron for DR classification. A cross-validation process is performed and high evaluation metrics are achieved. The whole system is then developed into a mobile application with a execution time of less than a second. The mobile application developed for the real-time screening of DR is presented by Shorav, Druzgalski, and Gautam in [27]; the application is based on the MobileNet architecture powered by Google TensorFlow.

III. DATASET AND MODEL IMPLEMENTATION

In this section, we will describe the implementation setup, outline the datasets and data preparation steps, and discuss the implementation of our models.

A. Experimental Setup

When it comes to the hardware used, Table I shows the specifications of the computer used to create the model.

TABLE I. COMPUTER HARDWARE.

CPU	Intel Core i7-9750H 2.60 GHz
RAM	32 GB
GPU	NVIDIA GeForce RTX 2070 8 GB

The software used for this project can be seen in the list below:

– Visual Studio Code for writing the Python code;

– Android Studio for Android emulators;

– Oracle VM VirtualBox for creating a Linux Virtual Machine to package the application into an APK file.

The project will consist of three stages: Data Preparation, Model Implementation, and Application Development.

B. Data Preparation

In this paper, we have used the Diabetic Retinopathy Dataset (https://www.kaggle.com/c/diabetic-retinopathy-detection/data). This is due to the large number of scans that the dataset includes for scans of healthy and unhealthy retinas, and this is needed for the classification models to learn the patterns needed for the classification.

Diabetic Retinopathy Dataset. The Diabetic Retinopathy Dataset was a dataset provided for a 2015 competition to generate a classification model that could detect diabetic retinopathy. There were five classes representing different stages of the disease, as well as healthy eyes. An interesting property of this dataset is that it combines the images of four diseased classes into one class and healthy images into another class; the number of images in both classes are the same (1831 images for each class, for a total of 3662 images). This means that there is no need for balancing. The problem this dataset had was that the resolution of the images was 1920×1080 pixels and as a result the dataset as a whole occupied 90 GB of storage. Therefore, all images were resized to 224×224 pixels dramatically reducing the size of the dataset and making it easier to work with. Lastly, the four stages of diabetic retinopathy were merged into one class called “diseased”.

C. Model Implementation

To compare various approaches, we have created four models, which will later be compared. We are going to create the models using the two aforementioned datasets, using two different architectures. As such, creating four pairs of dataset architecture.

MobilenetV3. This model uses MobilenetV3 as the base. The model is initialised using the pretrained weights from the ImageNet classifier to reduce the training time. On top of that, we have added additional layers to customise the model for the eye disease detection task. The following layers have been added.

– BatchNormalisation: This layer normalises the data at the mini-batch level, which allows us to use a high learning rate. It is known to improve ImageNet classification [28].

– Dense layer with ReLU activation, regularisation of L1 and L2, and dropout: This layer ensures that we avoid overfitting by adding a penalty term to the loss function. This results in the model having smaller weights [29].

– Output dense layer with softmax activation for classification: This layer is for prediction. It has two neurons because we have two classes that can be predicted. This layer takes the raw output values and transforms them into a probability distribution. Each neuron in the output layer represents the probability that the input belongs to a particular class.

Going into the hyperparameter tuning stage, we ended up setting the following parameters.

– Trainable - whether the base MobileNet model weights are trainable from the outset or not. If set to false, they become trainable after a set number of epochs. We set it to true because it can converge faster.

– Epochs - the number of training epochs, which determines how many times the whole dataset will be used to update the weights of the model. This is set to 100 so that there is enough time for the model to converge.

– Batch size - the number of images that are being taken for each iteration of training. This is set to 32 because it seems to yield the best results when compared to 16 and 8.

– Dropout rate - the number of neurons that are dropped out during each training step in the dropout layer. We set this to 0.1.

– Kernel regulariser: L2 regularisation applied to the weights of the dense layer. This is set to 0.16.

– Activity regulariser: L1 regularisation applied to the activation of the dense layer. This is set to 0.1.

– Bias regulariser: L1 regularisation applied to the bias terms of the dense layer. This is set to 0.1.

Custom Model. We have created a convolutional neural network (CNN) model from scratch where we extract features and look for important details, similar to how a real doctor would look at a scan and would look for certain anomalies. We will then try to find relationships between them using a neural network. The model has the following structure.

– Input layer: This is where the model receives the input data; in our case, the input data consist of images. Here we specify that we are expecting images of the 224×224 size.

– Convolutional layer 1: This layer scans the images received from the input and looks for patterns. It acts as a filter, learning to recognise shapes and edges. Here, we also transform all values into positive values using the “ReLU” activation.

– Max pooling layer 1: The output from the convolutional layers is usually too big; as such, it is common to reduce their sizes by taking the maximum value from a small region from the image. The point of this stage is to reduce the amount of computational power used.

– Convolutional layer 2: Similar to the first convolutional layer. The point of repeating this step is that each time we add another combination of convolutional layer + max pooling, we are looking for patterns using the simple patterns that were found in the previous layer. This allows the model to look at more complex patterns, since the features found by the layers become more complex the deeper we go.

– Max pooling layer 2: The pooling layers also have a generalising role, allowing us to see patterns, even though each image is different.

– Flatten layer: We take the output and transform it into a vector. This is a necessary step because the next layer takes a Vector as input, not a 2D grid.

– Dense layer 1: We feed the vector into a traditional

neural network layer. Here, we try to learn complex relationships between features. The network has 16 neurons.

– ReLU activation: We turn the output into positive numbers again.

– Dense layer 2: This layer is the prediction layer. It has two neurons matching the number of predicted classes. The output of this layer is a probability score for each class.

– Softmax activation: This layer ensures that our probabilities add up to 1 (100 %).

Once the training is done, we save a snapshot of the model so that it can be loaded into the mobile application and used as is. This is advantageous because it means that we are not limited by the hardware of mobile devices as we are not doing any training on the mobile device. We are using the model that is trained on the computer and deploying it on a mobile device.

IV. Application Development

This section discusses the technology stack that was used to build the mobile application, as well as the testing strategy that was followed.

A. Technology Stack

The application development stage had two important considerations that needed to be considered. The first one is to create an interface that allowed selecting a retina scan from their system, and the second is to package the application as an Android app. The Kivy framework was used to create the interface of the mobile application, including buttons and charts. TensorFlow Lite was used to deploy and run the machine learning models on the mobile device, and Buildozer was used to convert the Python code from the Kivy framework to Java for being deployable using Android Studio.

The mobile app was packaged as an Android Application Kit (APK) file, and this file can be used to install the mobile app on both physical devices and emulators, with a minimum Android API version of 24 and a maximum of 31.

B. Testing

The app was tested on two Android Studio emulators and two physical devices. We list the emulators and mobile phones that we used, respectively:

– An emulated Nexus One phone with Android Pie (API 28);

– An emulated Google Pixel with Android R (API 30);

– A Samsung Galaxy J5(2016) with Android Nougat (API 25);

– A Xiaomi Redmi 9T with Android Q (API 29).

The testing stage consisted of installing the application using the generated APK file, and then running the predictions back-to-back, using two eye scans available in the phone storage. The two images featured a healthy eye (Fig. 1) and a diseased eye (Fig. 2), respectively. Getting the correct results for both images using the application is considered a successful test. The screenshots below show the interface of the applications classifying healthy and diseased scans on two Android emulators.

Fig. 1. Screenshot of healthy eye scan on Nexus One emulator.

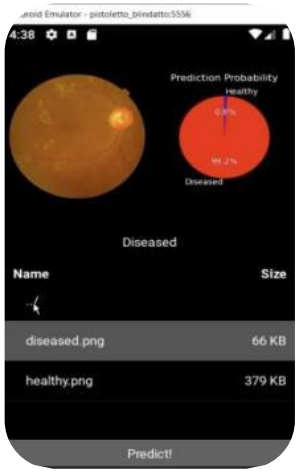

Fig. 2. Screenshot of the application identifying a scan with DR on Google Pixel emulator.Results and Discussion

C. Results

When looking at the models, there are various performance metrics that we have taken into account:

– Accuracy - measures how many predictions the model got right compared to the total number of predictions;

– Loss - a value that quantifies how far off the model predictions are from the actual correct values. Lower loss is better;

– F1 score - a metric that balances precision and recall. In our case, precision would tell us how often the model correctly identifies the disease when it says that it is positive. Recall would tell us how often the model correctly identifies the disease among all actual cases. The F1 score gives us a single number that considers both aspects.

These metrics have been tracked over the epochs to show how the model improved over time and where its performance plateaued.

Additionally, a confusion matrix has been created to look for model habits (e.g., a tendency to send many false positives).

Lastly, the models have been validated using K-fold cross-validation using five folds. These models have been trained on low batch sizes - 8 and epochs - 10 due to hardware limitations.

MobileNetV3. This model had a final accuracy value of 96.72 %. As shown in Fig. 3, it took around 20 epochs for the model to reach a 90 %+ accuracy. Figure 4 shows that the F1 score plateaus around epoch 25, which suggests that we could finish training early. Figure 5 shows that the model is relatively fast in learning, given the steep change in loss. Figure 6 shows the confusion matrix of the MobileNetV3 model on an unseen test dataset. This shows that the model is consistently accurate and can converge fast. The K-fold cross-validation is presented in Table II.

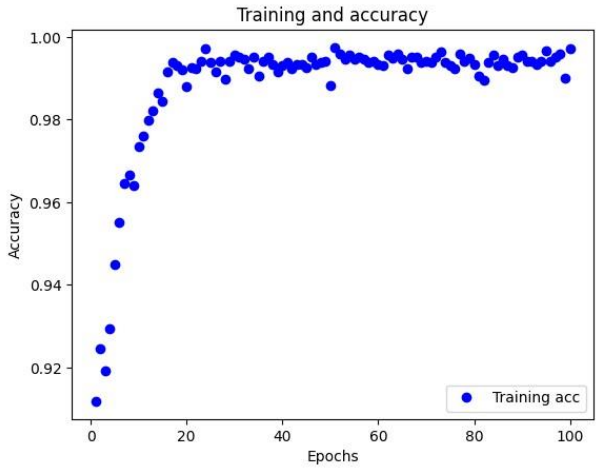

Fig. 3. MobileNetV3 - Accuracy over 100 epochs.

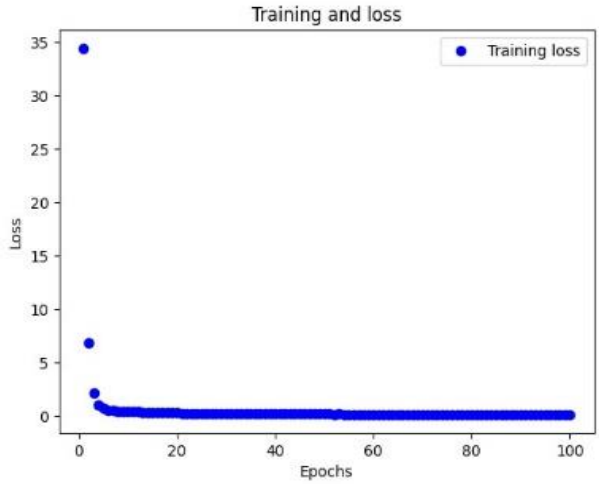

Fig. 4. MobileNetV3 - Loss metric over 100 epochs.

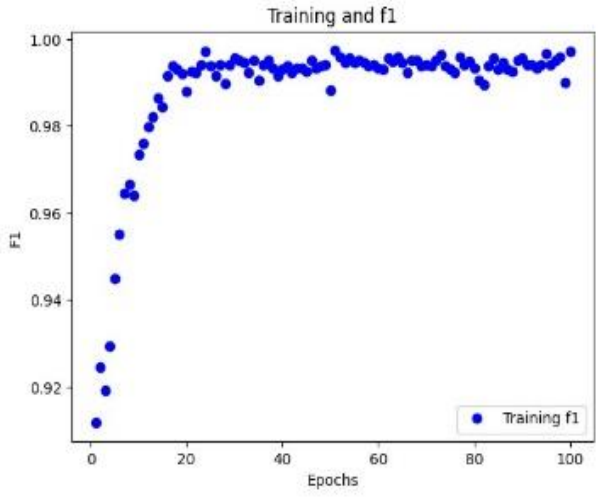

Fig. 5. MobileNetV3 - F1 metric over 100 epochs.

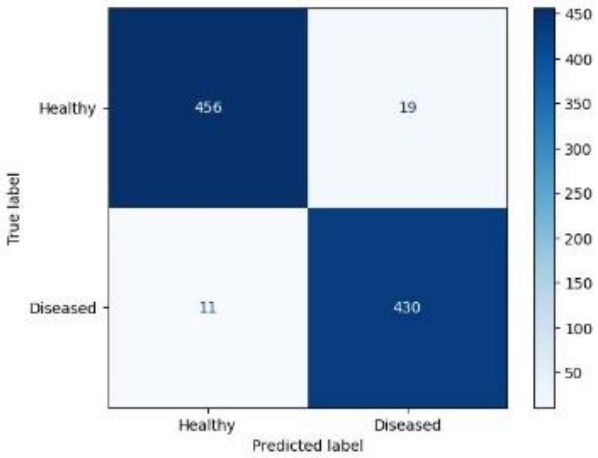

Fig. 6. MobileNetV3 - Confusion matrix.

TABLE II. THE K-FOLD CROSS-VALIDATION.

No.	MobileNetV3	Custom Model
1	0.97	0.48
2	0.95	0.91
3	0.93	0.94
4	0.95	0.48
5	0.97	0.95

Custom Model. This model had a final accuracy value of 93.45 %. Figure 7 shows a worrisome phenomenon: the model sometimes finds relationships between features when there are none.

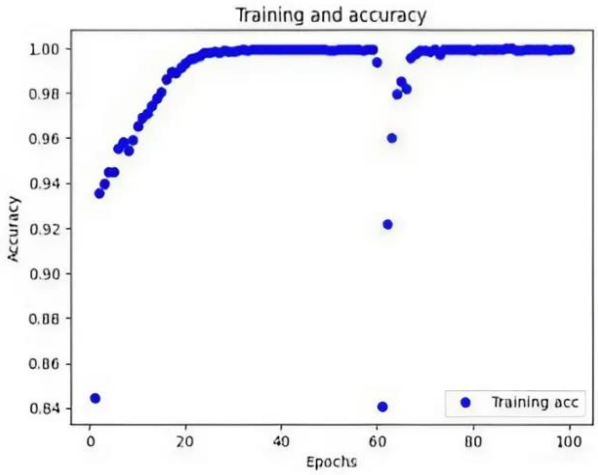

Fig. 7. Custom model - Accuracy over 100 epochs.

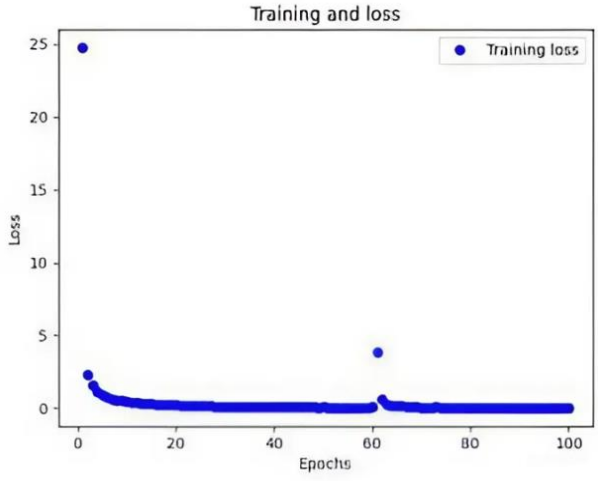

Fig. 8. Custom model - Model loss metric over 100 epochs.

This takes a long time to unlearn, and if this happens at the end of the training process, we might end up with a low quality model. The loss stray value in Fig. 8 presents this situation the best. The model took around 25 epochs to train according to Fig. 9, and Fig. 10 shows the confusion matrix of the custom model on an unseen test dataset.

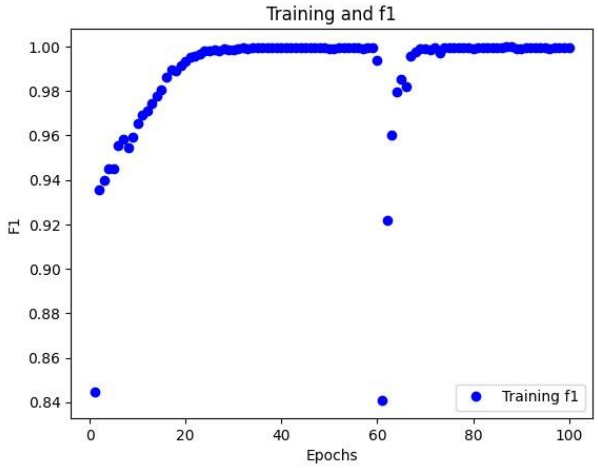

Fig. 9. F1 metric over 100 epochs of the custom model.

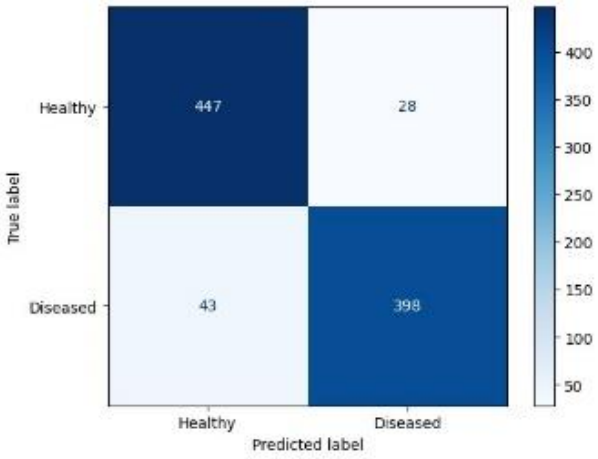

Fig. 10. Confusion matrix of the custom model.

The K-fold cross-validation results for the custom model are presented in the third column of Table II.

These values show that while the model usually has an accuracy above 90 %, sometimes its accuracy plummets to 50 %, which is a risk that must be taken into consideration. This translates to the fact that the model is inconsistent.

D. Discussion

Most of the literature on eye disease diagnosis that obtained improved performance applied deep neural network (DNN)-based approaches. These approaches are suitable for high performance computing that does not have any resource limitation. They do not reflect the constraint of the application based on embedded systems where lightweight models are needed for the implementation. So, we studied the best models, but emphasised that lightweight deep learning models should be used for segmentation as a back bone. The main reason is that shifting from HPC to an embedded platform does not allow us to compromise on the segmentation performance of the DNN model.

When comparing the models, it becomes clear why MobileNetV3 is such a staple when it comes to deep learning. Overall, the best model among the two produced was the one that used MobilenetV3. The use of K-fold cross-validation on a relatively large dataset (3662 images) suggests that the results that we obtained are likely to be representative. Further improvements can be obtained by training the models on a larger dataset. Continuous improvement of the models can be done after the application is deployed, by retraining the models and pushing the snapshots of the new models through updates of the application.

Based on our obtained classification results, we can say that MobileNetV3 is a suitable model that can be trained on HPC and used as a backbone in the mobile application. This is used to demonstrate the proof-of-concept. For the implementation of real applications, the authors suggest evaluating different DNN models and selecting a suitable one based on evaluation metrics. Then, such a model can be optimised for the mobile application development so that it is more specialised and lightweight at the same time.

V. CONCLUSIONS AND FUTURE WORK

In conclusion, this paper discussed the development of a lightweight convolutional neural network (CNN)-based model for retinal vessel segmentation and presented the development of an Android mobile application that uses the CNN model and is capable of loading an image from the user's phone and grade diabetic retinopathy. This paper showcases that mobile deep learning is indeed possible and can be used for the benefit of the end user. This work opens multiple avenues that can be explored to improve the developed model and the mobile app, e.g., 1) combining multiple datasets to create a larger one that can be used to generate more accurate results, 2) performing usability testing of the mobile app and thereby developing a more user-friendly interface.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] W. L. Alyoubi, W. M. Shalash, and M. F. Abulkhair, "Diabetic retinopathy detection through deep learning techniques: A review", *Informatics in Medicine Unlocked*, vol. 20, art. 100377, 2020. DOI: 10.1016/j.imu.2020.100377.

[2] A. Roy, D. Dutta, P. Bhattacharya, and S. Choudhury, "Filter and fuzzy c means based feature extraction and classification of diabetic retinopathy using support vector machines", in *Proc. of 2017 International Conference on Communication and Signal Processing (ICCSP)*, 2017, pp. 1844–1848. DOI: 10.1109/ICCSP.2017.8286715.

[3] D. J. Hemanth, O. Deperlioglu, and U. Kose, "An enhanced diabetic retinopathy detection and classification approach using deep convolutional neural network", *Neural Computing and Applications*, vol. 32, no. 3, pp. 707–721, 2020. DOI: 10.1007/s00521-018-03974-0.

[4] J. Mo, L. Zhang, and Y. Feng, "Exudate-based diabetic macular edema recognitionin retinal images using cascaded deep residual networks", *Neurocomputing*, vol. 290, pp. 161–171, 2018. DOI: 10.1016/j.neucom.2018.02.035.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", 2015. DOI: 10.1109/CVPR.2016.90.

[6] K. Aurangzeb, R. S. Alharthi, S. I. Haider, and M. Alhussein, "Systematic development of AI-enabled diagnostic systems for glaucoma and diabetic retinopathy", *IEEE Access*, vol. 11, pp. 105069–105081, 2023. DOI: 10.1109/ACCESS.2023.3317348.

[7] K. Aurangzeb, "A residual connection enabled deep neural network model for optic disk and optic cup segmentation for glaucoma diagnosis", *Science Progress*, vol. 106, no. 3, 2023. DOI: 10.1177/00368504231201329.

[8] L. Jain, H. V. S. Murthy, C. Patel, and D. Bansal, "Retinal eye disease

detection using deep learning", in *Proc. of 2018 Fourteenth International Conference on Information Processing (ICINPRO)*, 2018, pp. 1–6. DOI: 10.1109/ICINPRO43533.2018.9096838.

[9] A. K. Gangwar and V. Ravi, "Diabetic retinopathy detection using transfer learning and deep learning", in *Evolution in Computational Intelligence. Advances in Intelligent Systems and Computing*, vol. 1176. Springer, Singapore, 2021, pp. 679–689. DOI: 10.1007/978-981-15-5788-0_64.

[10] K. Aurangzeb, R. S. Alharthi, S. I. Haider, and M. Alhussein, "An efficient and light weight deep learning model for accurate retinal vessels segmentation", *IEEE Access*, vol. 11, pp. 23107–23118, 2022. DOI: 10.1109/ACCESS.2022.3217782.

[11] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation strategies from data", in *Proc. of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 113–123. DOI: 10.1109/CVPR.2019.00020.

[12] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast AutoAugment", in *Proc. of 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019, art. no. 598, pp. 6665–6675. DOI: DOI: 10.48550/arXiv.1905.00397.

[13] I. Goodfellow *et al.*, "Generative adversarial networks", *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2014. DOI: 10.1145/3422622.

[14] Y. Zhou, B. Wang, X. He, S. Cui, and L. Shao, "DR-GAN: Conditional generative adversarial network for fine-grained lesion synthesis on diabetic retinopathy images", *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 1, pp. 56–66, 2022. DOI: 10.1109/JBHI.2020.3045475.

[15] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, "Faster AutoAugment: Learning augmentation strategies using backpropagation", *Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science()*, vol. 12370. Springer, Cham, 2020, pp. 1–16. DOI: 10.1007/978-3-030-58595-2_1.

[16] Q. Zhang *et al.*, "Benchmarking of DL libraries and models on mobile devices", 2022. DOI: 10.48550/arXiv.2202.06512.

[17] M. Prajapati, S. K. Baliarsingh, J. Hota, P. P. Dev, and S. Das, "Retinal and semantic segmentation of diabetic retinopathy images using MobileNetV3", in *Proc. of 2023 International Conference on Computer, Electrical Communication Engineering (ICCECE)*, 2023, pp. 1–6. DOI: 10.1109/ICCECE51049.2023.10085191.

[18] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices", in *Proc. of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6848–6856. DOI: 10.1109/CVPR.2018.00716.

[19] M. Z. Alom *et al.*, "The history began from AlexNet: A comprehensive survey on deep learning approaches", 2018. DOI: 10.48550/arXiv.1803.01164.

[20] P. Nagda, M. Momaya, A. Pandey, A. Khanna, P. Verma, "Performance evaluation of various CNN network architectures for classification of diabetic retinopathy and normal retinal images", in *Soft Computing and Signal Processing. Advances in Intelligent Systems and Computing*, vol. 1325. Springer, Singapore, 2021, pp. 69–78. DOI: 10.1007/978-981-33-6912-2_7.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation", 2015. DOI: 10.1007/978-3-319-24574-4_28.

[22] K. Kipli *et al.*, "Development of mobile application for detection and grading of diabetic retinopathy", in *Proceedings of Trends in Electronics and Health Informatics. Lecture Notes in Networks and Systems*, vol. 376. Springer, Singapore, 2022, pp. 339–349. DOI: 10.1007/978-981-16-8826-3_29.

[23] S. Sheikh and U. Qidwai, "Using MobileNetV2 to classify the severity of diabetic retinopathy", *International Journal of Simulation: Systems, Science & Technology*, vol. 21, no. 2, pp. 16.1–16.6, 2020. DOI: 10.5013/IJSSST.a.21.02.16.

[24] I. Bidari, S. Chickerur, A. Kulkarni, A. Mahajan, A. Nikkam, and Abhishek THM, "Deploying machine learning inference on diabetic retinopathy in binary and multi-class classification", in *Proc. of 2021 International Conference on Industrial Electronics Research and Applications (ICIERA)*, 2021, pp. 1–6. DOI: 10.1109/ICIERA53202.2021.9726533.

[25] S. Bhatta, "Empowering rural healthcare: MobileNet-driven deep learning for early diabetic retinopathy detection in Nepal", *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 5, no. 4, pp. 290–302, 2023. DOI: 10.35882/jeeemi.v5i4.326.

[26] Y. Elloumi, N. Abroug, and M. H. Bedoui, "End-to-end mobile system for diabetic retinopathy screening based on lightweight deep neural network", *Advances in Intelligent Data Analysis XX. IDA 2022. Lecture Notes in Computer Science*, vol. 13205. Springer, Cham, 2022, pp. 66–67. DOI: 10.1007/978-3-031-01333-1_6.

[27] S. Suriyal, C. Druzgalski, and K. Gautam, "Mobile assisted diabetic retinopathy detection using deep neural network", in *Proc. of 2018 Global Medical Engineering Physics Exchanges/Pan American Health Care Exchanges (GMEPE/PAHCE)*, 2018, pp. 1–4. DOI: 10.1109/GMEPE-PAHCE.2018.8400760.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *Proc. of the 32nd International Conference on Machine Learning*, 2015, pp. 448–456.

[29] M. Yang, M. K. Lim, Y. Qu, X. Li, and D. Ni, "Deep neural networks with L1 and L2 regularization for high dimensional corporate credit risk prediction", *Expert Systems with Applications*, vol. 213, part A, art. 118873, 2023. DOI: 10.1016/j.eswa.2022.118873.