Automated detection of Diabetic Retinopathy using Deep Learning and deployment in smartphones

BT4813, Under Graduate Research Course

By

Anik Bhowmick

AE20B102

Under the guidance of

Nirav P. Bhatt

Interdisciplinary Dual Degree Data Science

Indian Institute of Technology Madras

Jan - Jul, 2024

ABSTRACT

The timely detection of Diabetic Retinopathy is crucial for preventing vision loss and improving patient outcomes. However, the analysis of fundus images can be resource-intensive and time-consuming, particularly in remote areas with limited access to trained personnel. This study aims to address these challenges by developing deep-learning-based approaches for the automated detection of diabetic retinopathy and other eye diseases from fundus images. Many state-of-the-art larger models such as VGG, and ResNet have shown promising performance in this challenge. However, these models are computationally heavier and remain an unsolved challenge for deployment in smartphones because of their limited computing capability. So in this paper, we will try to address this issue by using a lightweight mobile net v3 model, which has roughly 5.4M parameters and 225M FLOPS. We have achieved good results in some open-source datasets but the model is still struggling for some of the datasets which are affected by occlusion, refraction, variations in illumination, and blur. In this paper, we will discuss our data collection, processing, model training, and evaluation pipeline. It is still ongoing work, so the contents of this paper will change in the future.

CONTENTS

Chapter 1

INTRODUCTION

The diagnosis and management of Diabetic Retinopathy, a prevalent eye disease associated with diabetes, pose significant challenges to ophthalmologists. Timely and accurate identification of the disease's progression is critical for effective treatment and prevention of vision loss. With the increasing availability of digital imaging and the growing volume of ophthalmic data, there is a pressing need for advanced analysis techniques to extract fast meaningful insights from these datasets. Deep Learning serves as the root pillar for this purpose.

Deep learning models, such as ResNet, VGG, DenseNet, etc, have shown promising results in medical image analysis, including diabetic retinopathy detection. These models leverage their ability to automatically learn intricate features from ophthalmic images, enabling accurate classification and aiding in the early detection of the disease. By leveraging the power of these models, we aim to contribute to the growing body of knowledge on diabetic retinopathy diagnosis and facilitate the development of efficient screening tools for this condition. But As mentioned in the abstract in this paper we are interested in diving deeper into the use of the MobileNet V3 model for edge deployable devices. In our experiment, we utilized various open-source eye datasets such as APTOS, MESSIDOR, EYEPACS, IDRiD, ODIR, etc. These datasets contain images of Fundus having 3-4 different stages of Diabetic Retinopathy grading.

Chapter 2

LITERATURE REVIEW

With recent advancements in the field of AI, the availability of large datasets, and powerful hardware (GPU), AI techniques have been instrumental in making a lot of progress in the field of Ophthalmology. Some of the academic prior art that has made use of these technologies toward this goal are summarised below. Through this comprehensive literature review, we aim to identify the current trends, challenges, and potential future directions in the use of deep learning models for ophthalmology data analysis.

2.1 DEEP LEARNING FRAMEWORKS FOR DIABETIC RETINOPATHY DETECTION WITH SMARTPHONE-BASED RETINAL IMAGING SYSTEMS

Recep E. Hacisoftaoglu, Mahmut Karakaya, Ahmed B. Sallam et al.[1] experimented with various pre-trained models such as AlexNet, GoogLeNet, and ResNet50 on datasets like EyePACS, Messidor, Messidor-2, IDRiD, and University of Auckland Diabetic Retinopathy (UoA-DR). They used SGD with a learning rate of 1×10^{-5}, a momentum of 0.9, and a minibatch size of 8, 16, and 32 examples. The number of max epochs in experiments was set to 32, 64, and 128, depending on images in the training set. Their study also includes a comparison of the deep learning frameworks and to study of the effect of Field of view (FoVs) in smartphone-based retinal imaging systems on their DR detection accuracy. It was observed that the DR detection accuracy increases as the FoVs get larger and deep networks are trained with images from different datasets. They achieved the highest accuracy, sensitivity, and specificity of 98.6, 98.2, and 99.1 respectively in the test set for the ResNet50 Model.

2.2 AUTOMATED DETECTION AND CLASSIFICATION OF FUNDUS DIABETIC RETINOPATHY IMAGES USING SYNERGIC DEEP LEARNING MODEL

K. Shankar, Abdul Rahaman Wahab Sait, Deepak Gupta, S.K. Lakshmanaprabu, Ashish Khanna, Hari Mohan Pandey, et al.[2] applied Synergic Deep Learning (SDL) model was applied to classify the DR fundus images to various severity levels. The SDL consists of two separate Convolutional network systems called DCNN-A and DCNN-B the results obtained from these networks are passed to the fully connected layer for classification purposes. The justification for the presented SDL model was carried out on the Messidor DR dataset. The experimentation results indicated that the presented SDL model offers better classification than the existing models. From the experimental values, it is observed that the projected method exhibited excellent classification with the highest accuracy of 99.28, a sensitivity of 98.54, and a specificity of 99.38.

2.3 OUT OF DISTRIBUTION DETECTION AND ADVERSARIAL ATTACKS ON DEEP NEURAL NETWORKS FOR ROBUST MEDICAL IMAGE ANALYSIS

Anisie Uwimana, Ransalu Senanayake, et al[3]. focused on the Robustness of deep learning models, The robustness of deep learning algorithms needs to be evaluated before deploying them in the real world. Many deep learning model performs well on some data sets that came from the same distribution as the training data set. But they generalize very poorly on the data which came from out of distribution. In health care, AI OOD is quite common due to the usage of different instruments at different locations. They proposed the idea of a Mahalanobis distance-based confidence score. They implemented this in detecting abnormal input samples, in classifying malaria parasitized cells and uninfected cells. Their results indicated that the Mahalanobis confidence score detector exhibits improved performance and robustness of deep learning models, and achieved state-of-the-art performance on both out-of-distribution (OOD) and adversarial samples.

Chapter 3

METHODOLOGY

3.1 DATA ACQUISITION AND PREPROCESSING

The data for the first round of the experiment was collected from APTOS and MESSIDOR. Here is the statistics for the details of the data

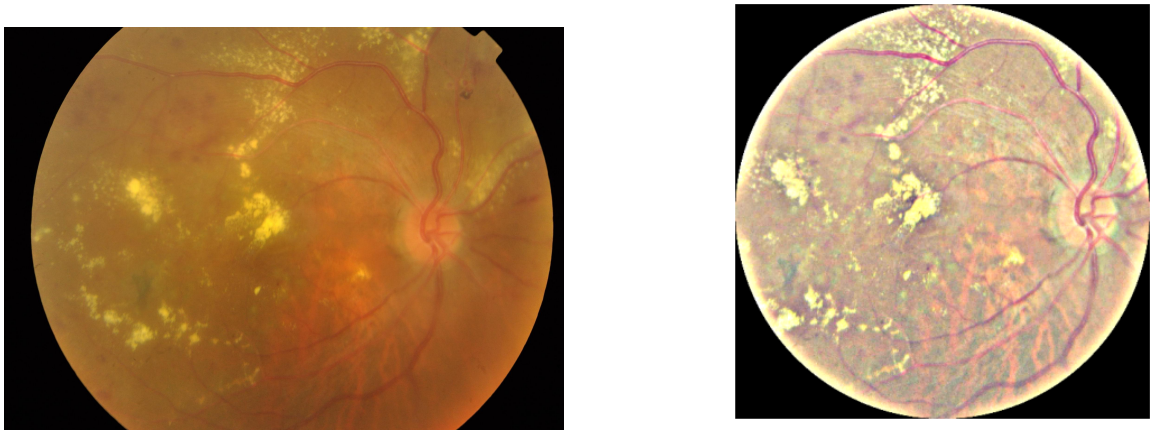
Diabetic Retinopathy Grade	**APTOS**	**MESSIDOR**
Normal (0)	1805	546
Grade 1 (1)	370	153
Grade 2 (2)	999	247
Grade 3 (3)	193	254
Grade 4 (4)	295	-

Table 3.1: Dataset details

So the number of images in APTOS is 3662 and MESSIDOR is 1200. Both are quite low in number for transfer learning. So we decided to augment the data. After preprocessing and augmentation, the number of data in the APTOS dataset grew to become 18310 and 10800 for the MESSIDOR. There were several issues in the images before the processing. The image processing step is summarized below:

- First, the images are cropped to contain only the fundus of the eye. This can be achieved by a technique called circle cropping. Because only the lighted part is the fundus of the eye in the image on a dark background, we can easily detect the sudden jump in the pixel values.

- It was found that some images were bright and some were dark. To avoid this dissimilarity, we applied Graham Ben's preprocessing. Here, the local average colour was subtracted. The local average was computed using Gaussian Blur.



(a) Before preprocessing (b) Image after preprocessing

Figure 3.1: Fundus image of eye comparison before and after preprocessing

- We applied random rotation onto the images to generate new images for the augmentation.

3.2 MODEL ARCHITECTURE

In health care problems, training a Neural network model from scratch by designing architecture is a very time-consuming process. So, we decided to use transfer learning.

3.2.1 MOBILENET V3

The model used for the first round of training is a MobileNetV3 network. We fine-tuned the model by replacing the fully connected top layers with an Attention module and dense networks. The primary use of this block is to extract the features. MobileNet V3 model comes with two variants: one is small architecture, and the other one is large. They differ by their number of parameters.

Model	MACs	Number of Parameters (M)	Top 1 Accuracy
MobileNet V3 Large	217	5.4	75.6
MobileNet V3 Small	66	2.9	68.1

Table 3.2: Model comparison (From TensorFlow)

The choice of MobileNet is of particular interest to us because it is easily deployable on smartphones. The architecture for the network is given below for reference.

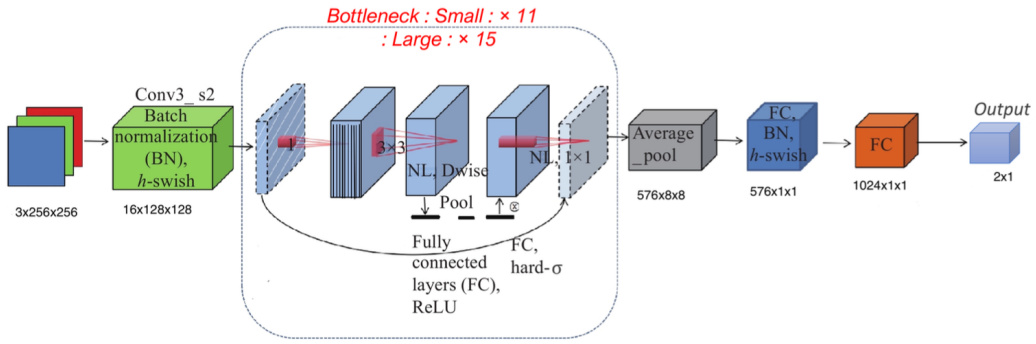


Figure 3.2: The architecture of small and large versions of MobileNet V3

One of the primary techniques employed is the use of depthwise separable convolutions. Instead of performing a standard convolution operation that involves convolving the input with a set of filters, MobileNet V3 separates the process into two steps: a depthwise convolution and a pointwise convolution. The depthwise convolution independently applies a single filter to each input channel. In contrast, the pointwise convolution projects the outputs of the depth-wise convolution to a new feature space using 1x1 convolutions. This separation significantly reduces the number of computations required compared to traditional convolutions. That's why it's ideal for deployment on mobiles. Additionally, MobileNet-V3 incorporates a range of activation functions to enhance inference efficiency further. These include hard-swish, sigmoid, and ReLU6. The hard-swish function, in particular, balances non-linearity and computational efficiency by combining linear and non-linear operations. This enables faster computation while still preserving the expressive power of the model. Moreover, MobileNet-V3 employs squeeze-and-excitation, which selectively enhances informative features by weighting channel-wise feature responses adaptively. This mechanism allows the network to focus on more relevant and discriminative features, improving accu-

racy and reducing computational requirements.[4]

3.2.2 ATTENTION BLOCK

The primary goal of incorporating an attention block is to increase the representation power of important features by suppressing non-important ones. As per the paper by Sanghyun Woo et al.[5] after extracting the features of the input from the last layer of the MobileNet V3 we fed it into the attention block where we sequentially applied channel and spatial attention so that each of the branches can learn 'what' and 'where' to attend in the channel and spatial axes respectively. The channel attention is computed by squeezing the spatial dimension and with the use of Average Pooling and Max Pooling in parallel. After the two pooling layers, the input is passed via shared MLP. After that, the two new feature vectors are elementwise added and passed through sigmoid activation to generate channel attention weight. This weight is then multiplied with the input and then sent for spatial attention. In spatial attention, the Average Pooling and Max Pooling are applied on the channel axis. The two new feature maps are then concatenated and a standard convolution followed by sigmoid activation is applied to generate the spatial attention weight. The following pictures will clarify how this technique works.

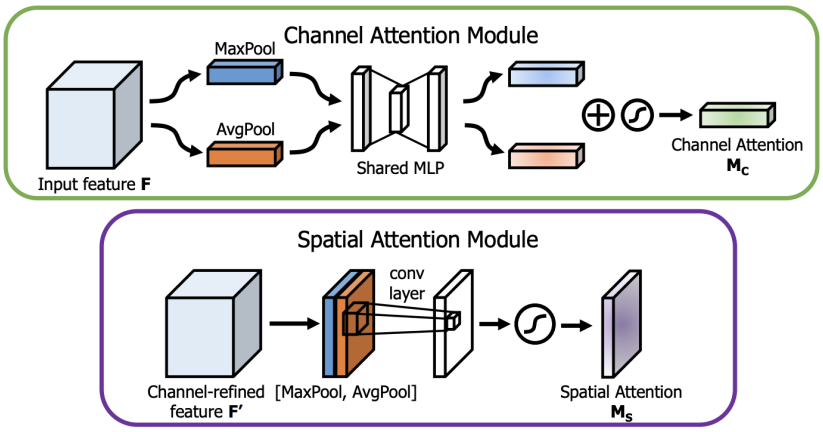


Figure 3.3: Convolutional Bottleneck attention module (source: Sanghyun Woo et al.[5])

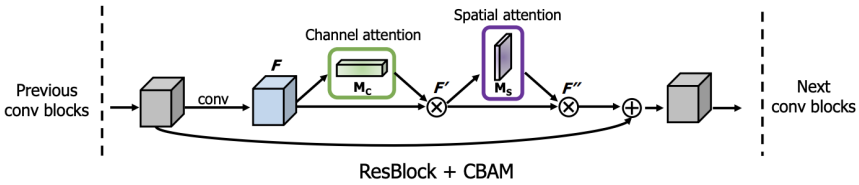


Figure 3.4: The Mechanism in flow (source: Sanghyun Woo et al.[5])

3.3 MOBILENET V3 LARGE WITH ATTENTION ARCHITECTURE

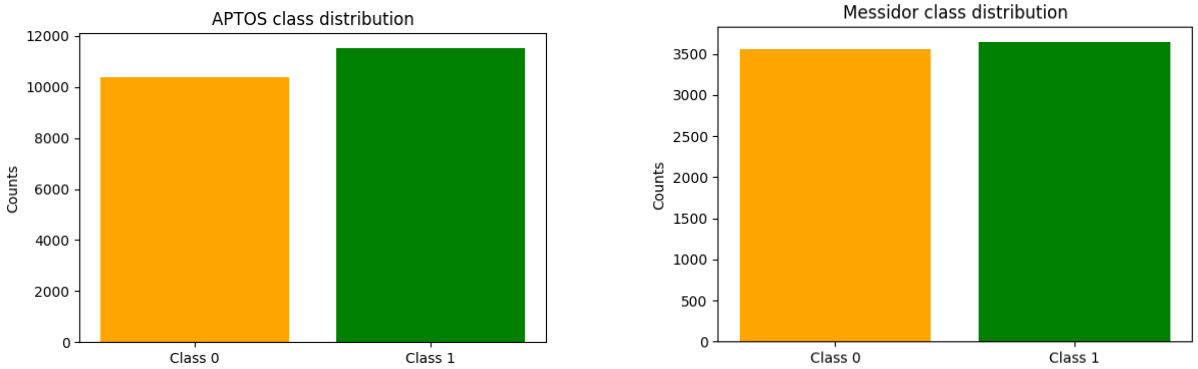
Below is the model summary and architecture used for our training purpose.

```
Model: "model"
_________________________________________________________________
Layer (type) Output Shape Param # Connected to
==================================================================================================
input_1 (InputLayer) [(None, 512, 512, 3)] 0 []

rescaling (Rescaling) (None, 512, 512, 3) 0 ['input_1[0][0]']

MobilenetV3large (Function (None, 16, 16, 960) 2996352 ['rescaling[0][0]']
al)

average_pooling2d (Average (None, 1, 1, 960) 0 ['MobilenetV3large[0][0]']
Pooling2D)

max_pooling2d (MaxPooling2 (None, 1, 1, 960) 0 ['MobilenetV3large[0][0]']
D)

flatten (Flatten) (None, 960) 0 ['average_pooling2d[0][0]',
'max_pooling2d[0][0]']

mlp_0_squeeze (Dense) (None, 240) 230640 ['flatten[0][0]',
'flatten[1][0]']

mlp_1 (Dense) (None, 960) 231360 ['mlp_0_squeeze[0][0]',
'mlp_0_squeeze[1][0]']

max_pool_add_avg_pool (Add (None, 960) 0 ['mlp_1[0][0]',
) 'mlp_1[1][0]']

sigmoid_channel (Activatio (None, 960) 0 ['max_pool_add_avg_pool[0][0]'
n) ]

multiply_20 (Multiply) (None, 16, 16, 960) 0 ['sigmoid_channel[0][0]',
'MobilenetV3large[0][0]']

tf.math.reduce_mean (TFOpL (None, 16, 16, 1) 0 ['multiply_20[0][0]']
ambda)

tf.math.reduce_max (TFOpLa (None, 16, 16, 1) 0 ['multiply_20[0][0]']
mbda)

combine_max_avg (Concatena (None, 16, 16, 2) 0 ['tf.math.reduce_mean[0][0]',
te) 'tf.math.reduce_max[0][0]']

conv2d (Conv2D) (None, 16, 16, 1) 99 ['combine_max_avg[0][0]']

sigmoid_spatial (Activatio (None, 16, 16, 1) 0 ['conv2d[0][0]']
n)

multiply_21 (Multiply) (None, 16, 16, 960) 0 ['sigmoid_spatial[0][0]',
'multiply_20[0][0]']

final_sum_att_feat (Add) (None, 16, 16, 960) 0 ['multiply_21[0][0]',
'MobilenetV3large[0][0]']

top_pool (GlobalAveragePoo (None, 960) 0 ['final_sum_att_feat[0][0]']
ling2D)

flatten_1 (Flatten) (None, 960) 0 ['top_pool[0][0]']

dropout (Dropout) (None, 960) 0 ['flatten_1[0][0]']

dense (Dense) (None, 64) 61504 ['dropout[0][0]']

dense_1 (Dense) (None, 2) 130 ['dense[0][0]']

==================================================================================================
Total params: 3520085 (13.43 MB)
Trainable params: 3495685 (13.33 MB)
Non-trainable params: 24400 (95.31 KB)
_________________________________________________________________
```

Figure 3.5: Summary of the model

3.4 TRAINING PROCEDURE

- **Class balancing:** We nearly balanced the two classes for each dataset due to preprocessing.



(a) Class distribution of APTOS data

(b) Class distributions of Messidor data

Figure 3.6: Distribution of classes

APTOS data	DR	No DR
Train	8293	7579
Validation	2071	1776
Test	1162	1029

Table 3.3: APTOS data details

Messidor data	DR	No DR
Train	2351	2289
Validation	566	555
Test	728	712

Table 3.4: Messidor data details

- **Optimizer selection:** We used a mini-batch size of 64 for both datasets. For optimizer, Adam was selected due to its various advantages, such as its ability to perform parameter-specific learning rate adaptation. A variety of learning rates was used, of which 1×10^{-4} for APTOS and 1×10^{-5} for Messidor data appeared most appropriate.

- **Regularization:** To prevent the model from overfitting, we used regularization techniques such as weight decay and dropouts; a standard dropout rate of 0.2 was chosen in the fully connected layers. We experimented with values as large as 0.1 to as small as 0.0001 for weight decay.

- **Loss function:** Because it's a binary classification problem, we used the Cross-Entropy function as our loss. Cross entropy is the ideal candidate for measuring deviation between two distributions.

- **Metrics selection:** Instead of relying on the accuracy for evaluating the model performance, we used precision, recall and F_1 score to assess the model performance on the test sets thoroughly.

- **Hardware selection:** Because deep learning tasks are computationally expensive, training on the CPU takes quite a long time. So, we used 2 Tesla T4 GPUs from Kaggle for our experiment and performed distributed training. It made the training sufficiently faster.

Chapter 4

RESULTS AND DISCUSSIONS

4.1 MOBILENET V3 SMALL EXPERIMENT

We first trained the mobile net v3 small model on the APTOS data. The Model achieved high train accuracy (99%) and validation accuracy (98%) within 10 epochs. We evaluated the model thoroughly on the APTOS test set of size 2191 by checking F_1 scores for each class.

APTOS data	**Accuracy (%)**	**Precision**		**Recall**		**F_1 score**	
		No DR	**DR**	**No DR**	**DR**	**No DR**	**DR**
Train	99.45	0.98	0.99	0.99	0.99	0.99	0.99
Test	98.17	0.97	0.99	0.99	0.97	0.98	0.98

Table 4.1: APTOS training and testing details after 10 epochs.

Next, we trained a fresh model on the Messidor data. The model struggled to converge here, so we needed to train for some 60 epochs.

Messidor data	**Accuracy (%)**	**Precision**		**Recall**		**F_1 score**	
		No DR	**DR**	**No DR**	**DR**	**No DR**	**DR**
Train	99.11	0.98	1.00	1.00	0.98	0.99	0.99
Test	89.17	0.84	0.96	0.97	0.82	0.90	0.88

Table 4.2: Messidor training and testing details after 60 epochs.

The model performance for the test set is not good, indicating a probable overfitting.

4.2 MOBILENET V3 LARGE EXPERIMENT

Because the small network was struggling with the Messidor dataset. So, we trained the Messidor data on MobileNet V3 large network.

Messidor data	**Accuracy (%)**	**Precision**		**Recall**		**F_1 score**	
		No DR	**DR**	**No DR**	**DR**	**No DR**	**DR**
Train	99.97	0.99	0.99	1.00	0.99	0.99	0.99
Test	93.88	0.93	0.96	0.96	0.92	0.94	0.94

Table 4.3: Messidor training and testing details after 30 epochs.

So, now it's clear that the large model can handle the challenging dataset Messidor.

4.3 INTERPRETABILITY USING SALIENCY MAP

Because of the black-box nature of deep learning models, the explainability of these models always remains a highly sought-after research interest. It's important to know what pixels are exactly firing for the model to make decisions for predicting class labels. This helps to assess the model. For explainability, we used the saliency map, which is actually computed by taking the gradient of the loss function with respect to the input image pixel. This results in a newer feature map where each pixel essentially holds the gradient value of the loss. We then superimposed it on the original image to highlight those pixels, which triggered the model to make the decision.

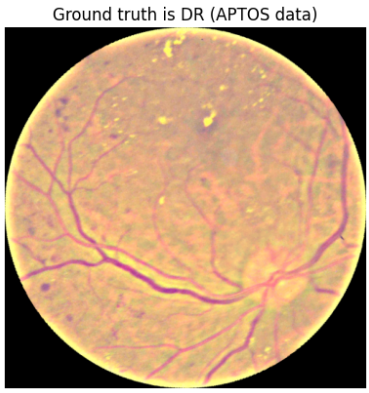


Figure 4.1: Original DR image of APTOS data

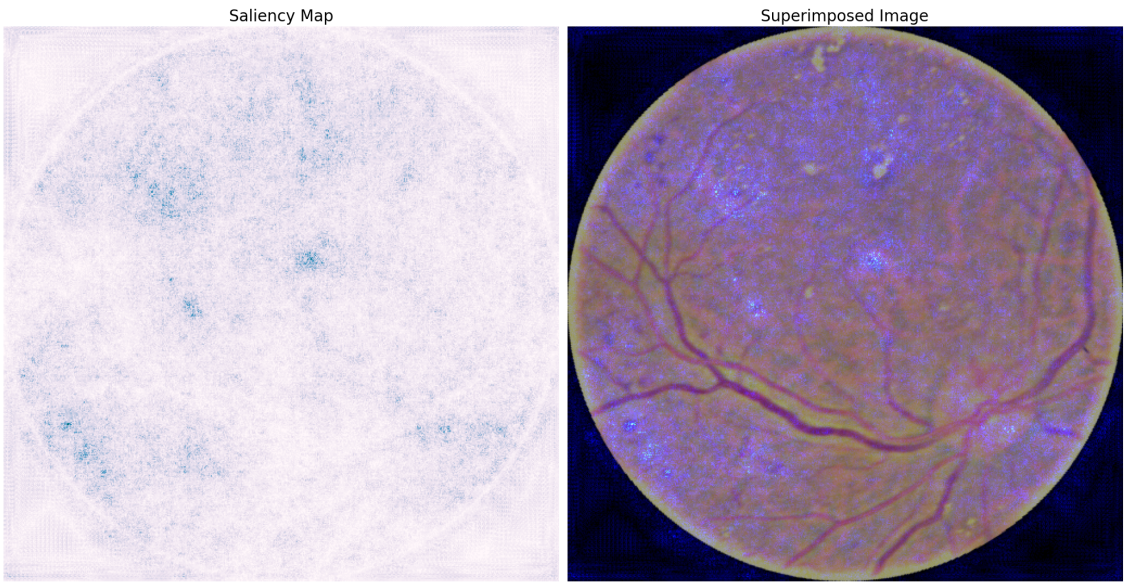


Figure 4.2: Saliency map

Figure 4.3: Original DR image of Messidor data

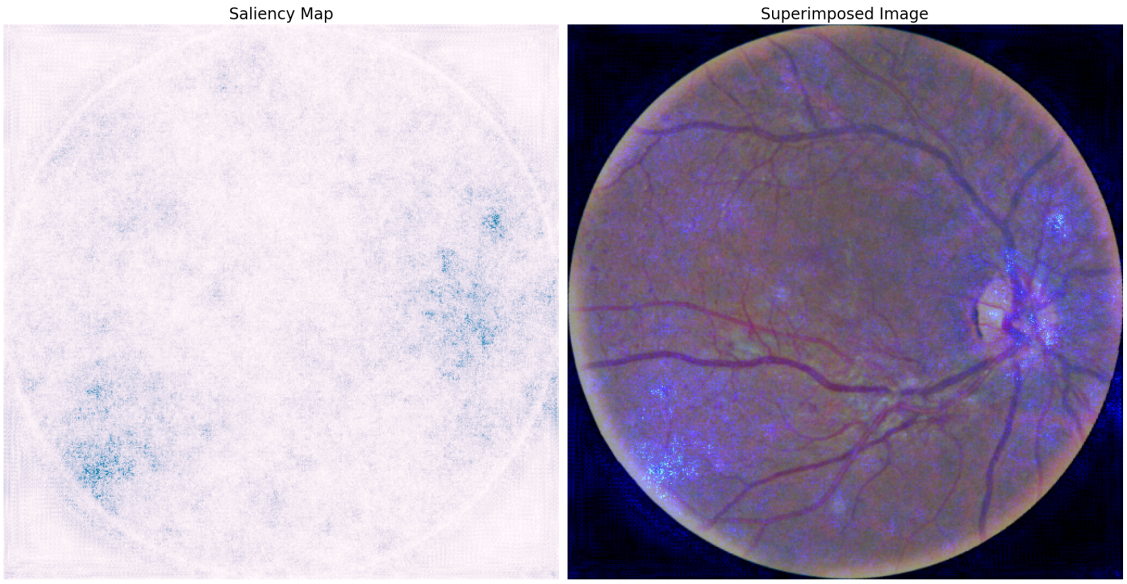


Figure 4.4: Saliency map

Chapter 5

CONCLUSION

This brings us to the end of our first round of experiments using MobileNet architecture. MobileNet model achieved sufficiently good results for these two datasets. However, we will try to improve this performance even more on the Messidor Dataset in our next round of experiments using Vision transformers.

BIBLIOGRAPHY

[1] R. E. Hacisoftaoglu, M. Karakaya, and A. B. Sallam, "Deep learning frameworks for diabetic retinopathy detection with smartphone-based retinal imaging systems," 2020. [Online]. Available: https://sciencedirect.com/science/article/pii/S016786552030129X

[2] K. Shankar, A. R. W. Sait, D. Gupta, S. Lakshmanaprabu, A. Khanna, and H. M. Pandey, "Automated detection and classification of fundus diabetic retinopathy images using synergic deep learning model," 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865520300714

[3] A. Uwimana and R. Senanayake, "Out of distribution detection and adversarial attacks on deep neural networks for robust medical image analysis," 2021. [Online]. Available: https://arxiv.org/pdf/2107.04882.pdf

[4] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019. [Online]. Available: https://arxiv.org/pdf/1905.02244v5.pdf

[5] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," 2018. [Online]. Available: https://arxiv.org/pdf/1807.06521.pdf