Eye Disease Detection

Harshkumar Patel
C0848343
Artificial Intelligence and Machine Learning, Lambton College, Toronto

Kishan Lakhatariya
C0849341
Artificial Intelligence and Machine Learning, Lambton College, Toronto

Mohammaddin Masbi
C0850500
Artificial Intelligence and Machine Learning, Lambton College, Toronto

Kanan Trivedi
C0851492
Artificial Intelligence and Machine Learning, Lambton College, Toronto

Deekshith Pothedar
C0851255
Artificial Intelligence and Machine Learning, Lambton College, Toronto

***Abstract*— Eye diseases affect millions of people worldwide and can lead to significant vision impairment or even blindness if left undetected and untreated. Early detection of eye diseases is crucial in preventing permanent vision loss and improving patient outcomes. Traditional methods of diagnosing eye diseases often require specialized equipment and trained professionals, which could be costly and time taking.**

In recent years, advances in machine learning and deep learning have provided new opportunities for automated diagnosis of eye diseases. Many studies have investigated the use of retinal images for the diagnosis of eye diseases, but less attention has been given to external eye images. External eye images provide a non-invasive and easily accessible source of information for eye disease diagnosis.

The aim of our project is to evaluate the performance of our proposed deep learning model in the automated detection of cataract eye diseases using external eye images. Ultimately, this project has the potential to assist healthcare professionals in the early detection and prevention of various eye diseases, ultimately improving patient outcomes.

***Keywords*— eye detection, data exploration, data preprocessing, training, testing, evaluation metrics**

I. INTRODUCTION

Eye diseases are a significant cause of visual impairment and blindness worldwide, affecting millions of people. Early detection and timely treatment are essential for preventing vision loss and improving outcomes for patients. However, diagnosing eye diseases can be challenging, and the accuracy of diagnosis often depends on the expertise of the clinician.

There is a lot of potential for strategies to increase the precision and speed of medical diagnosis using Artificial intelligence (AI) and machine learning (ML) techniques. Recently, there has been a very major increase in the development of AI/ML systems for medical imaging analysis, including the detection of eye diseases.

This project aims to develop an AI/ML system that can accurately detect cataract eye diseases from external images. We have used ResNet-50 and Efficient Net machine learning algorithms to analyze images of the eye and identify any abnormalities or signs of disease.

We have trained a model on a large dataset of labelled images that includes normal eyes and eyes with the disease. The dataset is preprocessed and augmented to get a better accuracy model.

After training the model, we've integrated it into a basic web application that can take input images of the eye and provide a diagnosis. The application will be user-friendly and accessible to both medical professionals and patients.

The primary goal of our project is to improve the accuracy and speed of diagnosing cataract eye diseases, which can lead to earlier treatment and better outcomes for patients. In underserved areas where there may be a dearth of ophthalmologists, this system may also assist in easing the pressure on healthcare professionals and increasing access to care.

II. RELATED STUDY

1. "Automated detection of diabetic retinopathy using deep learning" by Gulshan et al. This study trained a deep learning algorithm to recognize diabetic retinopathy using a dataset of over 100,000 retinal pictures. The algorithm's detection of referable diabetic retinopathy has a sensitivity of 90.3% and a specificity of 98.1%.

2. "Automated screening for age-related macular degeneration in photographic retina images using deep learning" by Ting et al. This study was able to detect age-related macular degeneration with an accuracy of 94.5% on a dataset of over 5,000 images using a deep-learning algorithm

3. "Automated detection of glaucoma using deep learning" by Christopher et al. This study used a dataset of over 16,000 retinal images to train a deep learning algorithm to detect glaucoma. The algorithm achieved an accuracy of 85.3% and a sensitivity of 87.5% for detecting glaucoma.

III. METHODOLOGY

A. *Data Exploration:*

The size and composition of the dataset are important factors to consider when developing an AI/ML model. It is quite important to ensure that the dataset is large enough and diverse enough to capture the full range of eye diseases that the model is designed to detect. We have used a manual dataset for our project. The size of the whole dataset is 3549 with which the size of normal eye image data is 2684 and the size of cataract eye image data is 865.

Data Exploration

```
In [2]: cataract = os.path.join(
            "Datasets/raw dataset/Cataract",
            random.choice(os.listdir("Datasets/raw dataset/Cataract")))
        normal = os.path.join("Datasets/raw dataset/Normal",
                              random.choice(os.listdir("Datasets/raw dataset/Normal")))
        fig, ax = plt.subplots(1, 2)

        ax[0].imshow(cv2.cvtColor(cv2.imread(cataract), cv2.COLOR_BGR2RGB))
        ax[1].imshow(cv2.cvtColor(cv2.imread(normal), cv2.COLOR_BGR2RGB))

Out[2]: <matplotlib.image.AxesImage at 0x17e425730>
```

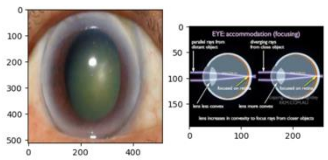

Figure 1: Data Exploration

B. *Data Preprocessing:*

The important stage of our project is to pre-process, to ensure that the images are in a format that is suitable for analysis by the AI/ML model. This may involve resizing, normalization, and augmentation of the images to improve the accuracy of the model.

Pre-Processing

```
In [3]: rf = Roboflow(api_key="dHg9mKRRDSCBTlbaEl8w")
        project_eye = rf.workspace().project("eyes-dpurk")
        model_eye = project_eye.version(2).model
        project_iris = rf.workspace().project("iris_120_set")
        model_iris = project_iris.version(7).model

        loading Roboflow workspace...
        loading Roboflow project...
        loading Roboflow workspace...
        loading Roboflow project...

In [4]: def extract_eye(img):
            eye_predictions = model_eye.predict(img, confidence=20,
                                                overlap=40).json()['predictions']
            eyes = []
            if eye_predictions != []:
                for prediction in eye_predictions:
                    if prediction["class"] == "Open Eyes":
                        X, Y, W, H = int(prediction['x']), int(prediction['y']), int(
                            prediction['width']), int(prediction['height'])

                        H = int(H / 2)
                        W = int(W / 2)

                        cropped_image = img[Y - H:Y + H, X - W:X + W]
                        eyes.append(cropped_image)

            return eyes

In [5]: def extract_iris(img):
            predictions_iris = model_iris.predict(img, confidence=40,
                                                  overlap=40).json()['predictions']
            iris = None
            if predictions_iris != []:
                for prediction in predictions_iris:
                    if prediction["class"] == "0":
                        flag = False
                        X, Y, W, H = int(prediction['x']), int(prediction['y']), int(
                            prediction['width']), int(prediction['height'])

                        H = int(H / 2)
                        W = int(W / 2)

                        iris = img[Y - H:Y + H, X - W:X + W]

            return iris
```

Figure 2: Data preprocessing

C. *Dataset Splitting:*

In this part, we have split the whole dataset into two parts which are training and testing. We have used a manual dataset for our project. For training, 2083 files have been used, and for validation, 520 files have been used.

Splitting Up Dataset

```
In [10]: height, width = 120, 120
         training_batch_size = 32

In [11]: train_set = tf.keras.preprocessing.image_dataset_from_directory(
             "Datasets/train",
             validation_split=0.2,
             subset="training",
             seed=21,
             image_size=(height, width),
             batch_size=training_batch_size)

         Found 2603 files belonging to 2 classes.
         Using 2083 files for training.

In [12]: image_cat = train_set.class_names
         print(image_cat)

         ['Cataract_iris', 'Normal_iris']

In [13]: validation_set = tf.keras.preprocessing.image_dataset_from_directory(
             "Datasets/train",
             validation_split=0.2,
             subset="validation",
             seed=21,
             image_size=(height, width),
             batch_size=training_batch_size)

         Found 2603 files belonging to 2 classes.
         Using 520 files for validation.
```

Figure 3: Dataset Splitting

D. *Model Training and Testing:*

1. Efficient Net: EfficientNet is a family of neural network architectures that was proposed by Tan and Le in 2019. It is designed to be both accurate and efficient, with a focus on achieving the best performance on image classification tasks while minimizing the count of parameters and computational resources required. The EfficientNet architecture is based on a novel scaling method that systematically scales the network depth, width, and resolution. This scaling allows the model to be efficiently trained on datasets of varying sizes and complexities. The EfficientNet family includes several models, each with a different scaling factor. EfficientNet could be used to develop a highly accurate and efficient model for detecting a range of eye diseases.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb7 (Functional)  (None, 2560)             64097687

 module_wrapper (ModuleWrapp  (None, 128)              327808
 er)

 module_wrapper_1 (ModuleWra  (None, 64)               8256
 pper)

 module_wrapper_2 (ModuleWra  (None, 32)               2080
 pper)

 module_wrapper_3 (ModuleWra  (None, 1)                33
 pper)

=================================================================
Total params: 64,435,864
Trainable params: 64,125,137
Non-trainable params: 310,727
_________________________________________________________________

In [16]: efficientnet.compile(optimizer=Adam(learning_rate=0.001),
                              loss='binary_crossentropy',
                              metrics=['accuracy'])

In [19]: history = efficientnet.fit(train_set, validation_data=validation_set, epochs=10)

         Epoch 1/10
         66/66 [==============================] - 58s 827ms/step - loss: 0.0226 - accuracy: 0.9923 - val_loss: 0.0737
         - val_accuracy: 0.9808
         Epoch 2/10
```

Figure 4: Training Efficient Net

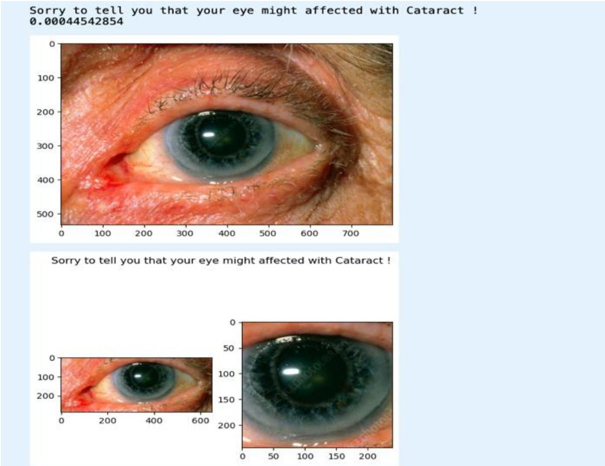

Figure 5: Testing Efficient Net

2. ResNet-50: Microsoft Research introduced the ResNet-50 deep convolutional neural network architecture in 2015. It is a variation of the ResNet architecture, which is renowned for its capacity to circumvent the degradation issue in deep neural networks, and has 50 convolutional neural networks (CNN) layers. In the context of eye disease detection from external images, ResNet-50 could be used as a backbone for a deep learning model that could accurately classify images of various eye diseases. The ResNet-50 architecture's ability to learn complex features from images could be leveraged to identify subtle differences between images and make accurate diagnoses.

```
In [37]: history = dnn_model.fit(train_set, validation_data=validation_set, epochs=10)
Epoch 1/10
2023-04-07 20:41:14.931511: W tensorflow/core/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency
: 0 Hz
2023-04-07 20:41:14.931831: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin o
ptimizer for device_type GPU is enabled.
66/66 [==============================] - ETA: 0s - loss: 0.1583 - accuracy: 0.9342
2023-04-07 20:41:19.886353: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin o
ptimizer for device_type GPU is enabled.
66/66 [==============================] - 7s 93ms/step - loss: 0.1583 - accuracy: 0.9342 - val_loss: 0.1056 - val_a
ccuracy: 0.9558
Epoch 2/10
66/66 [==============================] - 4s 64ms/step - loss: 0.0540 - accuracy: 0.9822 - val_loss: 0.0942 - val_a
ccuracy: 0.9654
Epoch 3/10
66/66 [==============================] - 4s 65ms/step - loss: 0.0411 - accuracy: 0.9837 - val_loss: 0.0887 - val_a
ccuracy: 0.9654
Epoch 4/10
66/66 [==============================] - 4s 65ms/step - loss: 0.0217 - accuracy: 0.9918 - val_loss: 0.0679 - val_a
ccuracy: 0.9788
Epoch 5/10
66/66 [==============================] - 4s 65ms/step - loss: 0.0147 - accuracy: 0.9952 - val_loss: 0.1967 - val_a
ccuracy: 0.9385
Epoch 6/10
66/66 [==============================] - 4s 65ms/step - loss: 0.0258 - accuracy: 0.9904 - val_loss: 0.1329 - val_a
ccuracy: 0.9654
Epoch 7/10
66/66 [==============================] - 4s 65ms/step - loss: 0.0061 - accuracy: 0.9981 - val_loss: 0.0993 - val_a
ccuracy: 0.9769
Epoch 8/10
66/66 [==============================] - 4s 65ms/step - loss: 0.0023 - accuracy: 0.9995 - val_loss: 0.1007 - val_a
ccuracy: 0.9750
Epoch 9/10
66/66 [==============================] - 4s 65ms/step - loss: 6.2323e-04 - accuracy: 1.0000 - val_loss: 0.1055 - v
al_accuracy: 0.9788
Epoch 10/10
66/66 [==============================] - 4s 66ms/step - loss: 3.7017e-04 - accuracy: 1.0000 - val_loss: 0.1083 - v
al_accuracy: 0.9808
```

Figure 6: Training ResNet-50

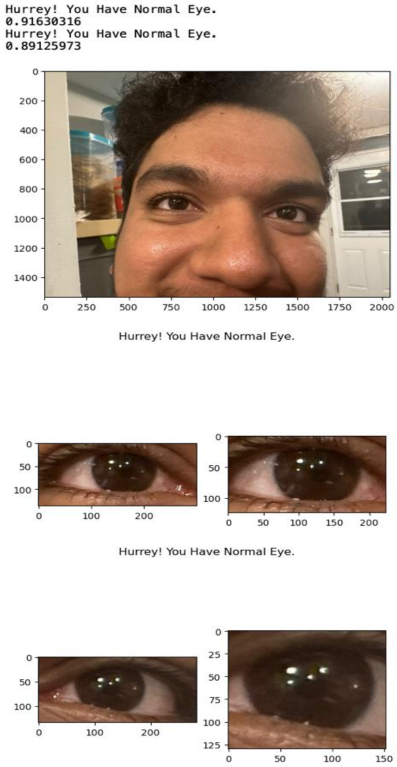

Figure 7: Testing ResNet-50

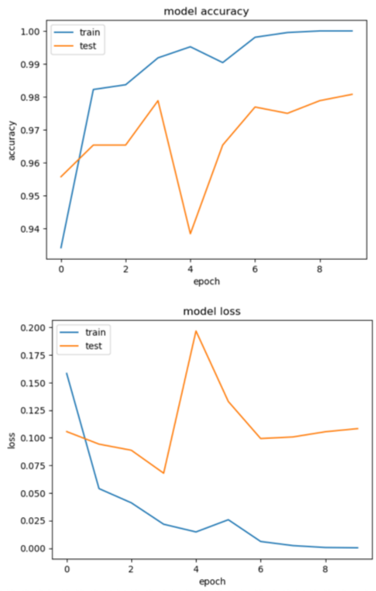

Figure 8: Accuracy and Loss Graph

IV. FINAL RESULTS

Here, we have compared Efficient Net and ResNet-50. Efficient Net is better as we got better accuracy for Efficient Net. Also, the detection is better in Efficient Net compared to ResNet-50.

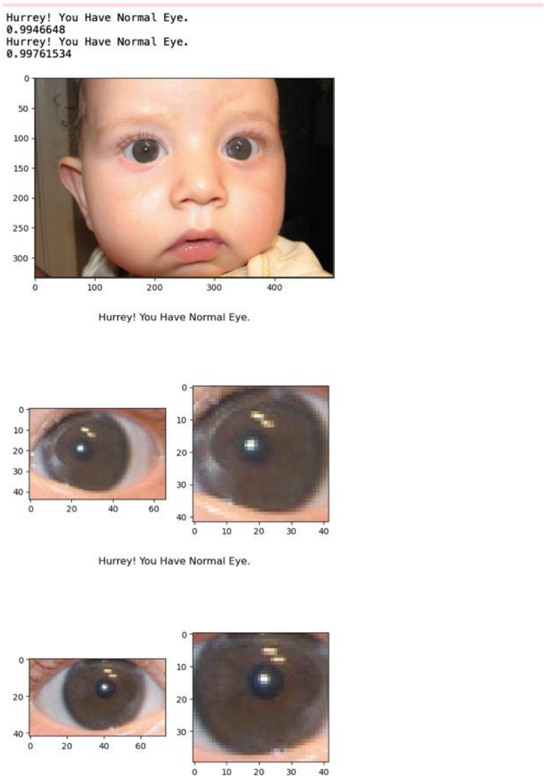

Figure 9: Output of Efficient Net

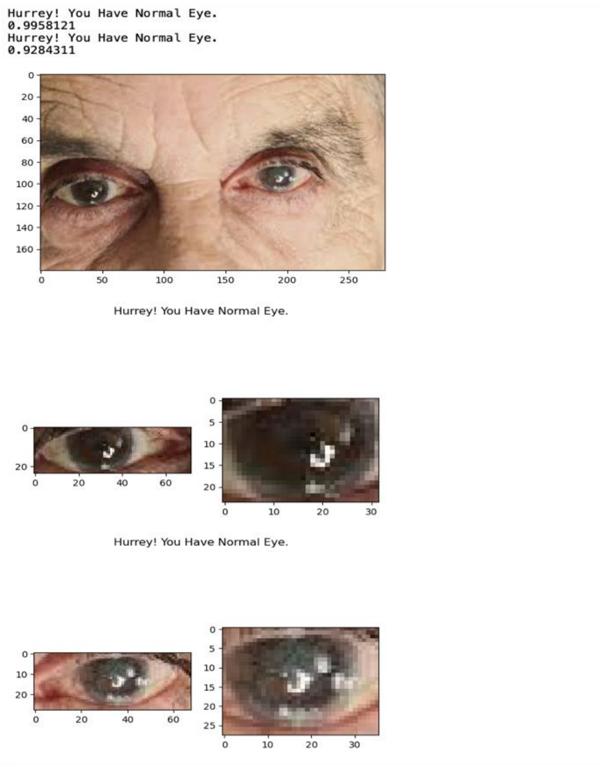

Figure 10: Output of ResNet-50

V. EVALUATION METRICS

A genuine positive indicates that the model accurately predicted a positive class, whereas a false positive indicates that the model did the opposite. Checking True Positive, True Negative, False Positive, and False Negative values from the dataset is the main goal of a confusion matrix.

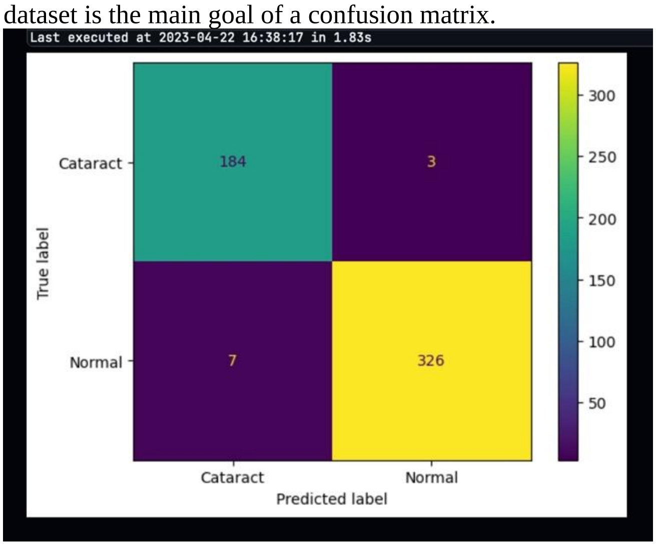

Figure 11: ResNet-50

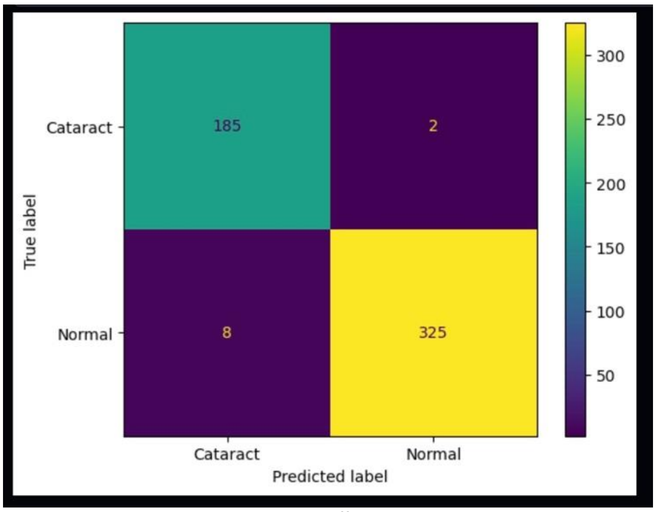

Figure 12: Efficient Net

VI. FRONT-END

We have used Flask to develop basic web application for our model. We have created a basic web application. There is a login page, registration page, input page, and result page in this application. It is user-friendly.

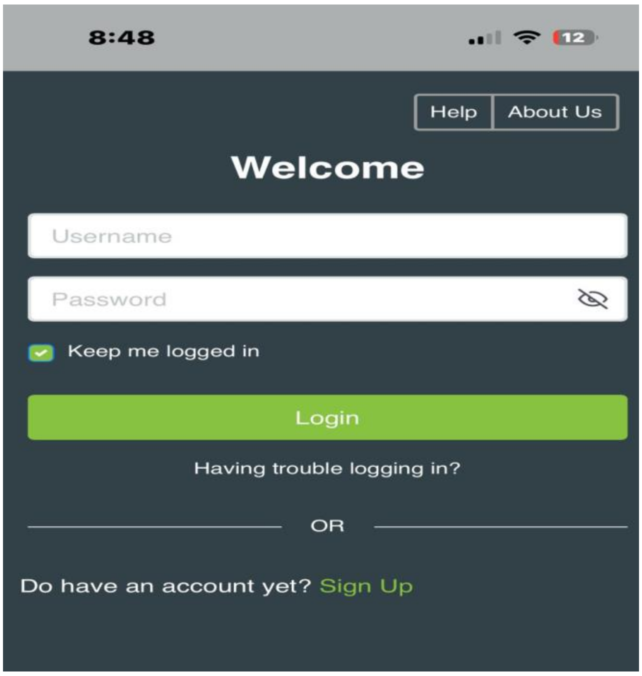

Figure 13: Login Page

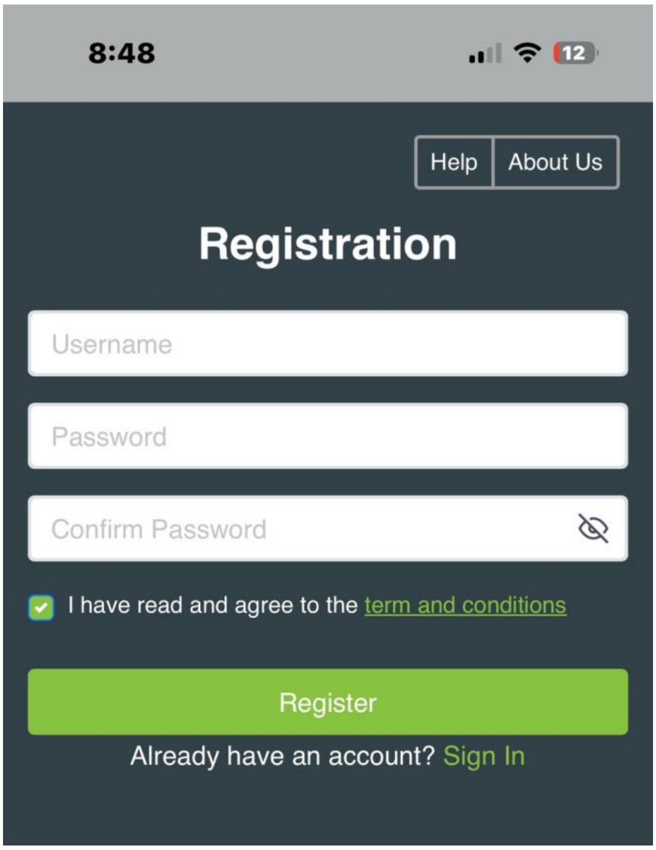

Figure 14: Registration Page

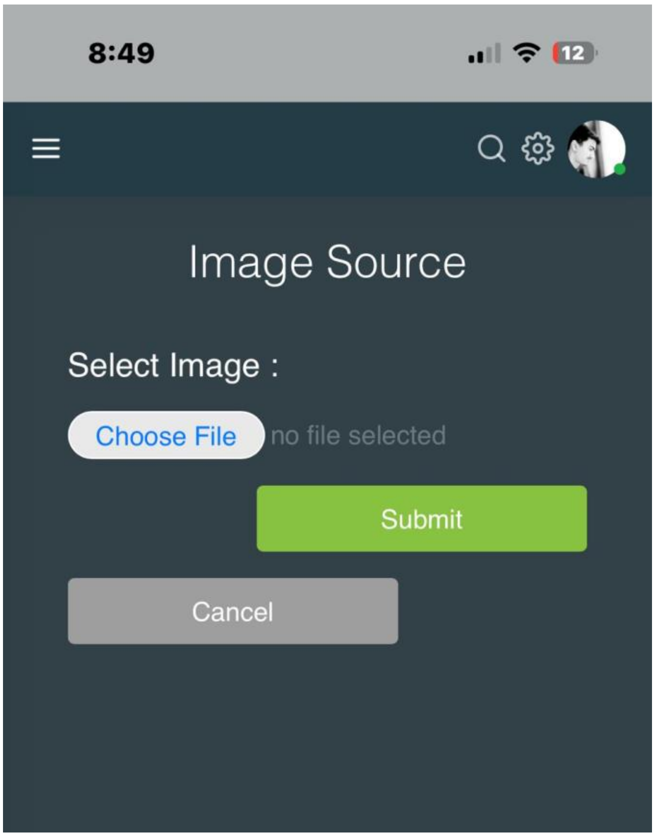

Figure 15: Input Page.1

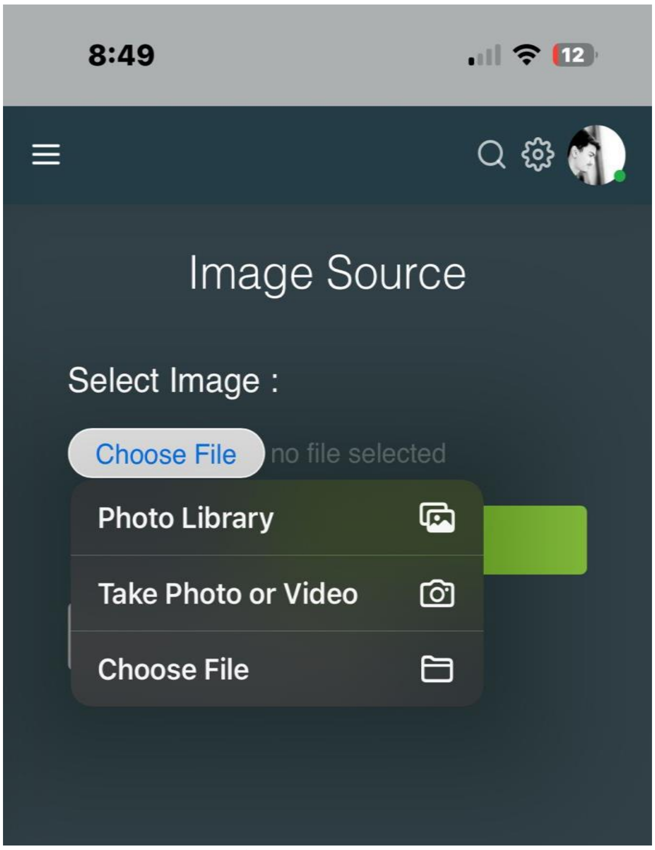

Figure 16: Input Page.2

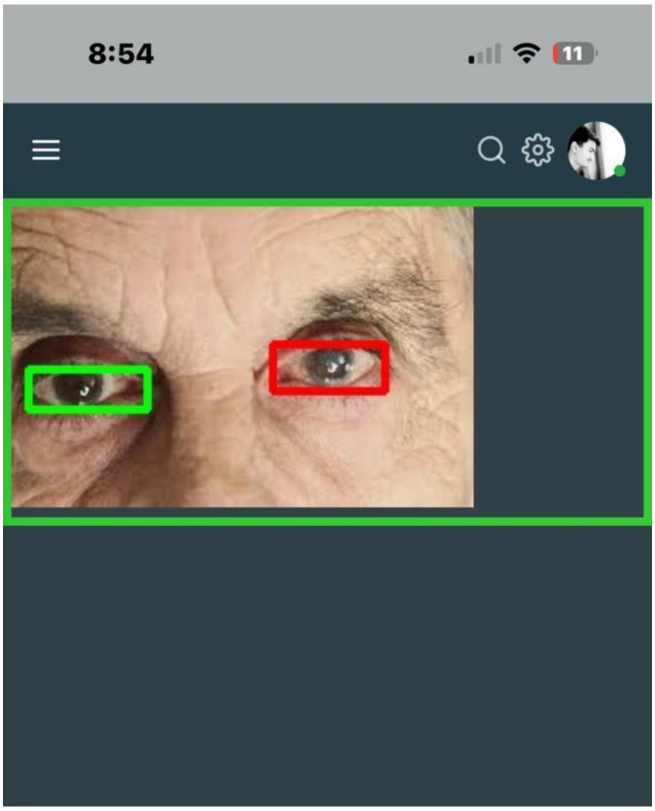

Figure 17: Result Page

VII. CONCLUSION

In conclusion, there is tremendous potential for improving the precision and speed of diagnosis, which can result in better patient outcomes, through the creation of an AI/ML system for the detection of cataract eye illnesses from exterior photographs. The use of Efficient Net and ResNet-50 machine learning algorithms has shown promising results in detecting eye disease.

The success of such a system depends on the availability of high-quality labelled datasets, the development of accurate and efficient models, and the integration of these models into user-friendly applications that can be widely adopted in clinical settings. Further research and development in this field will help improve the accuracy, generalizability, and accessibility of these systems, ultimately leading to better patient care and outcomes.

VIII. FUTURE WORK

There is a lot of potential for future work in the development of an AI/ML system for the detection of eye diseases from external images. Here are a few areas where further research and development could be valuable:

1. Expansion of the dataset: While existing datasets provide a solid foundation for training AI/ML models, expanding the dataset to include very wider range of eye diseases and a more diverse population would improve the accuracy and generalizability of these models.

2. Transfer learning: One method for fine-tuning a pre-trained model for a particular job, such the identification of a specific eye illness, is transfer learning. This technique could be used to improve the accuracy of models trained on smaller datasets.
3. Integration with electronic health records (EHRs): Integrating AI/ML systems for the detection of eye diseases with EHRs would allow for more seamless and efficient patient care, as well as more comprehensive tracking and analysis of patient data.
4. Deployment in clinical settings: The successful deployment of AI/ML systems for the detection of eye diseases in clinical settings will require collaboration between clinicians, researchers, and developers, as well as careful consideration of issues related to patient privacy, data security, and regulatory compliance.

Overall, continued research and development in this field will help to advance the accuracy, accessibility, and effectiveness of AI/ML systems for the detection of eye diseases from external images, ultimately leading to better patient care and outcomes.

IX. REFERENCES

[1] A global review of publicly available datasets for ophthalmological imaging: barriers to access, usability, and generalizability
https://www.thelancet.com/journals/landig/article/PIIS2589-7500(20)30240-5/fulltext

[2] Detection of signs of disease in external photographs of eyes via deep learning
https://www.nature.com/articles/s41551-022-00867-5

[3]
https://www.mathworks.com/help/deeplearning/ref/resnet50.html#

[4]
https://medium.com/mlearning-ai/understanding-efficientnet-the-most-powerful-cnn-architecture-eaeb40386fad