Machine Learning Approach to Detect & Annotate Eye Diseases using Retinal Images

2023-162

Our Team

Team Leader
IT20166106

Member 01
IT20165666

Member 02
IT20227890

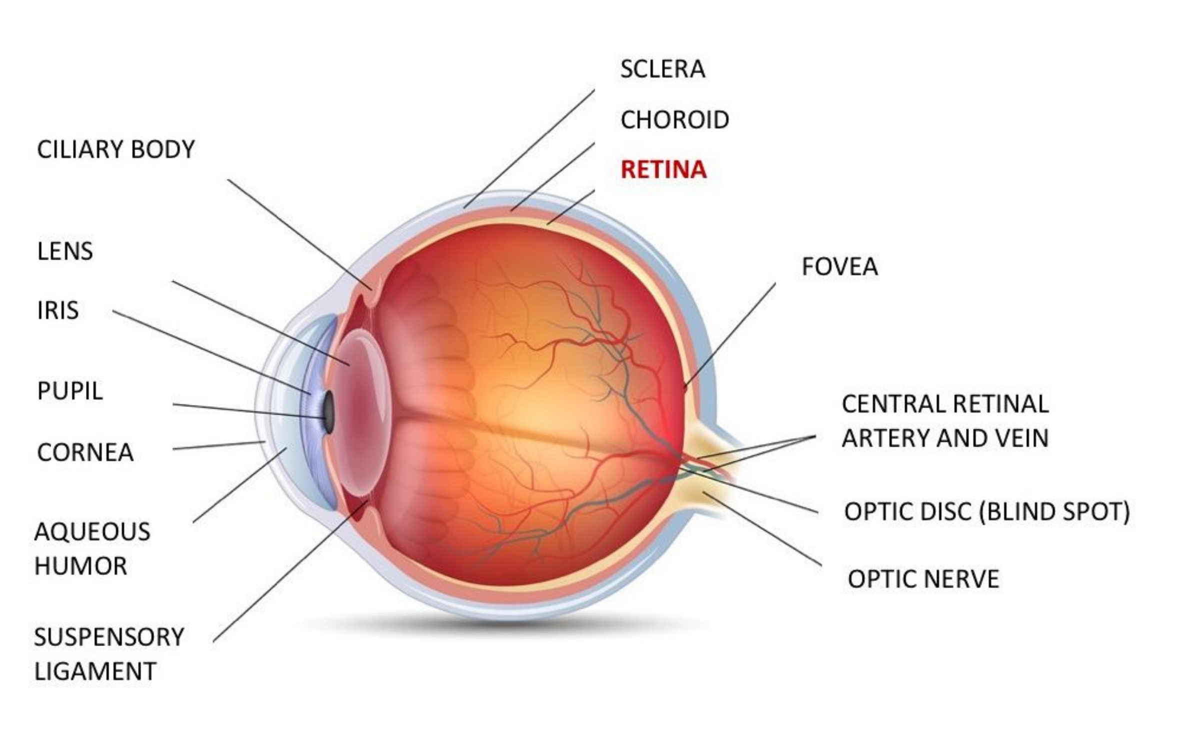
Member 03
IT20172978

Introduction

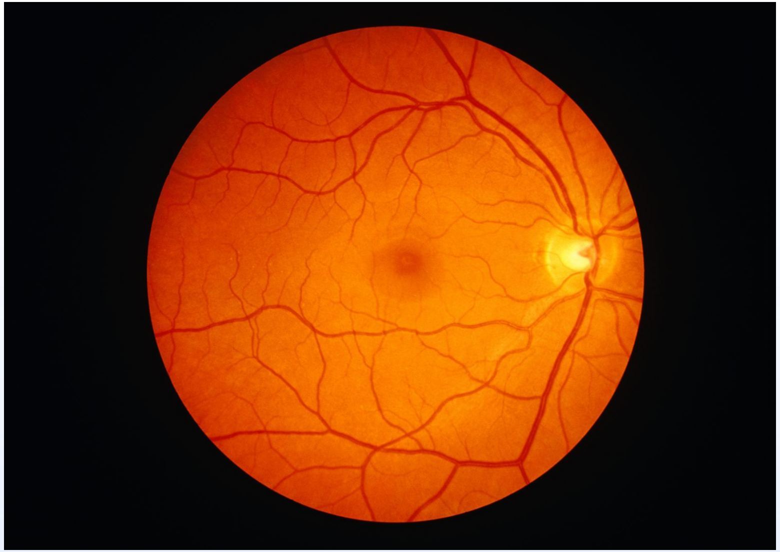
- Development of an mobile application for the identification of eye diseases like diabetic retinopathy and age-related macular degeneration to aid eye specialists with reliable and accurate diagnoses.

- Detection of eye diseases using retinal images, grade their severity and classify according to their types.

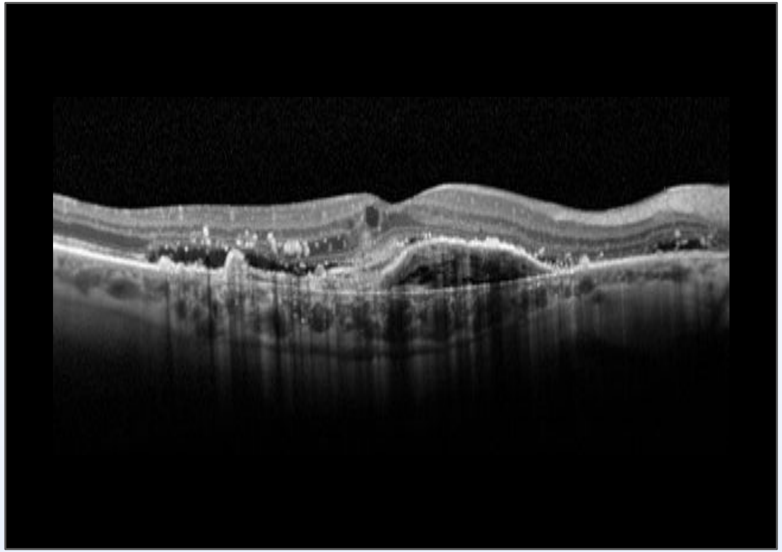
What is Retina



Types of Retinal Images



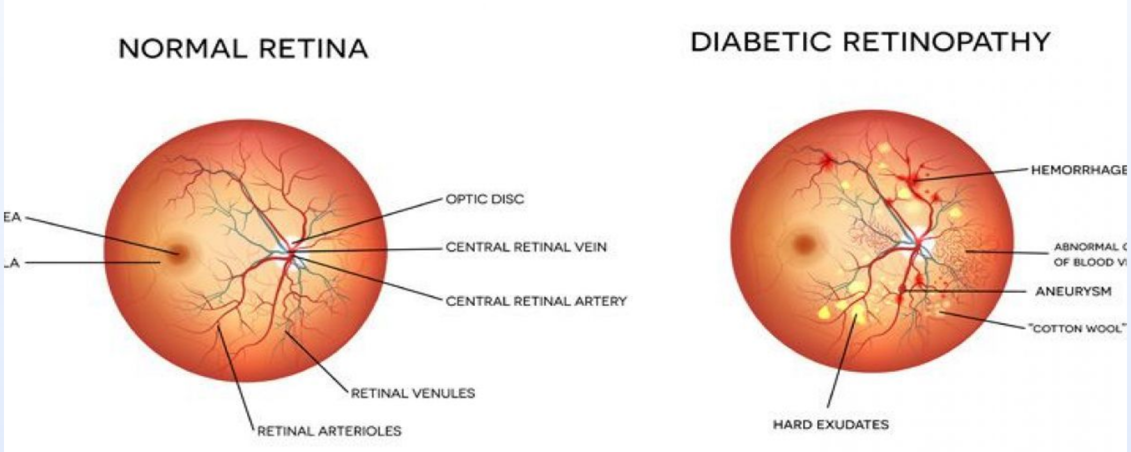
Fundus Image



OCT Image

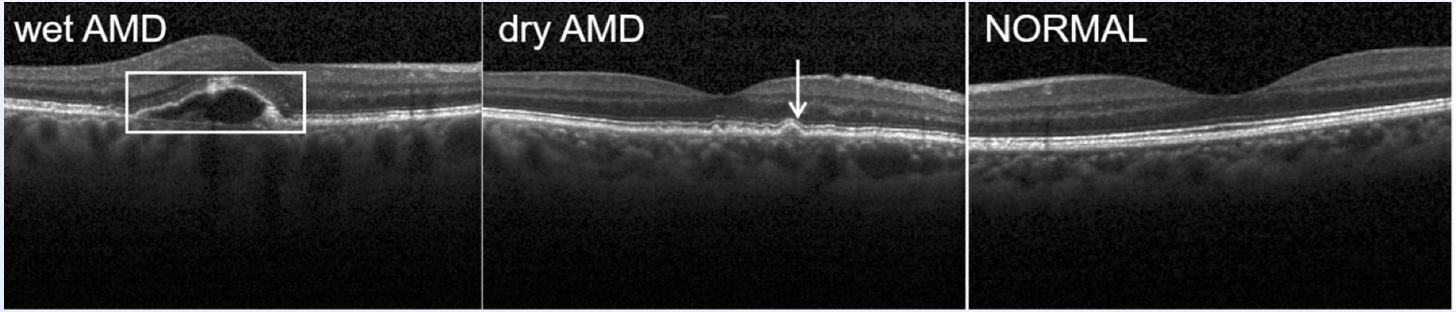
What is Diabetic Retinopathy

- Diabetic retinopathy is a complication of diabetes
- Diabetic retinopathy is the 4th leading cause of blindness and 5th common cause of visual impairment



What is Age-related Macular Degeneration

- Age-Related Macular Degeneration (AMD) causes progressive vision loss and affects millions worldwide.
- There are two types of AMD:
  - Dry AMD (Drusen)
  - Wet AMD (CNV)



Research Objectives

- Improve accuracy and efficiency of the outcomes for better diagnosis
- Early diagnosis for patients
- Reduce workload for eye specialists in order to focus on critical situations
- Save time and resources of medical field

Commercialization

- Target market for our mobile application includes healthcare institutions, clinics, hospitals and ophthalmologists
- Unique Selling Proposition
  - Our mobile app offers a fast and accurate solution for diabetic retinopathy detection
  - Unavailability of verified mobile applications for the purpose

Commercialization

- Business model for our application is through a subscription-based and one-time in-app purchase
- Distribution of the application is done by publishing on verified app marketplaces
- Future Expansion
    - Continuous improvements
    - Expansion of functionalities

IT20166106 | Perera H. A. N. S

Detect Symptoms of Diabetic Retinopathy using Retinal Fundus Images

Introduction

- Diabetic retinopathy is a leading cause of blindness
- Early detection and timely intervention are crucial for preventing vision loss in diabetic patients.
- Conventional methods for diabetic retinopathy screening are time-consuming and require manual interpretation by trained specialists.

Research Problem

- Being able to optimize outcomes for a mobile application
- Screening process of DR requires lots of resources
- Ability to detect DR without any assistance of an opthamologist
- Ability to improve the accuracy

Main Objectives

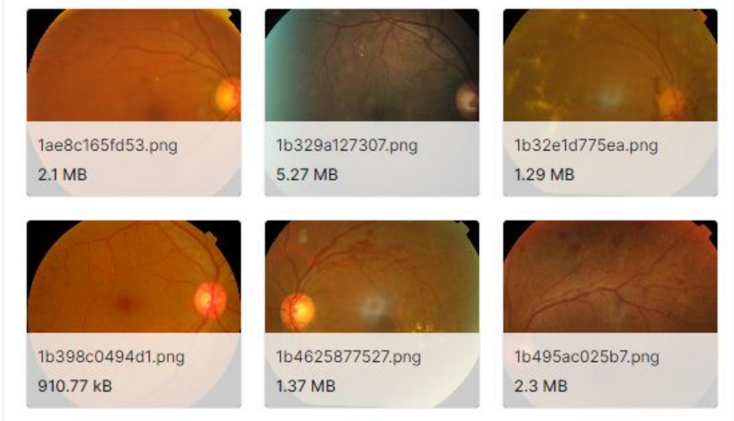
- Develop a deep learning algorithm to accurately detect the presence of Diabetic Retinopathy in a retinal fundus image to aid the screening process of diagnosis

Sub Objectives

- Preprocess retinal fundus image
- Extract features of the retina
- Preparation of datasets & development of the deep learning model
- Evaluate the model performance
- Identify the diagnosis of DR

Datasets

- Asia Pacific Tele-Ophthalmology Society 2019 Blindness Detection (APTOS 2019 BD) dataset
- 3662 samples of retinal fundus images
- https://www.kaggle.com/c/aptos2019-blindness-detection/overview



Implementation of Component

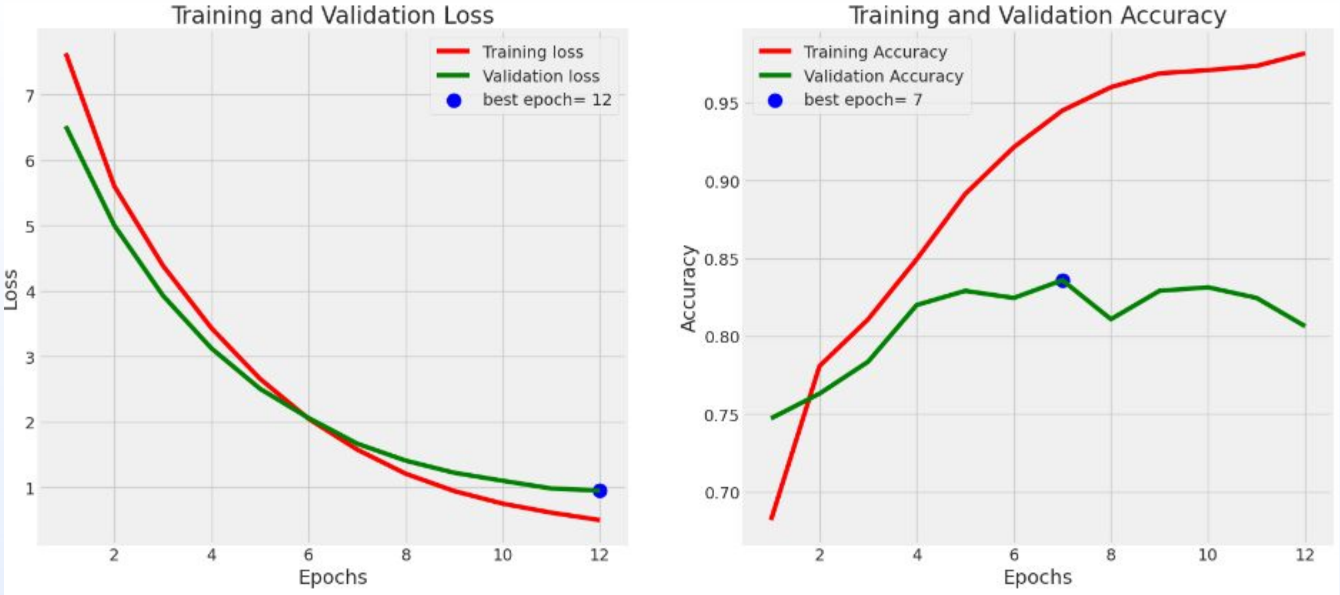
- Frameworks and Libraries
  - Tensorflow
  - keras
- Model Architecture Summary
  - EfficientNetB3
  - BatchNormalization
  - Dense
  - Dropout
- Loss Function and Optimization
  - Adam

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb3 (Functional)  (None, 1536)             10783535

 batch_normalization (BatchN  (None, 1536)             6144
 ormalization)

 dense (Dense)               (None, 256)               393472

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 5)                 1285

=================================================================
Total params: 11,184,436
Trainable params: 11,094,061
Non-trainable params: 90,375
_________________________________________________________________
```

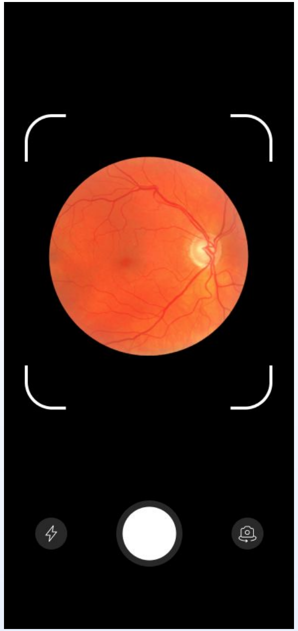
Result Comparison

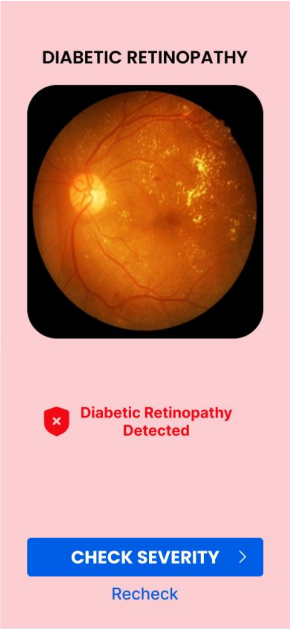
Epoch	Loss	Accuracy	V_loss	V_acc	LR	Next LR	Monitor	% Improv	Duration
1 /40	7.634	68.180	6.51997	74.715	0.00100	0.00100	accuracy	0.00	129.90
2 /40	5.602	78.081	5.00284	76.310	0.00100	0.00100	accuracy	14.52	45.96
3 /40	4.389	81.086	3.93151	78.360	0.00100	0.00100	accuracy	3.85	46.50
4 /40	3.433	84.944	3.12381	82.005	0.00100	0.00100	accuracy	4.76	47.67
5 /40	2.658	89.143	2.50608	82.916	0.00100	0.00100	accuracy	4.94	48.10
6 /40	2.048	92.147	2.05986	82.460	0.00100	0.00100	val_loss	17.81	46.43
7 /40	1.578	94.503	1.66799	83.599	0.00100	0.00100	val_loss	19.02	46.82
8 /40	1.208	96.005	1.40822	81.093	0.00100	0.00100	val_loss	15.57	46.24
9 /40	0.943	96.893	1.22289	82.916	0.00100	0.00100	val_loss	13.16	46.80
10 /40	0.751	97.098	1.10002	83.144	0.00100	0.00100	val_loss	10.05	47.11
11 /40	0.613	97.371	0.98263	82.460	0.00100	0.00100	val_loss	10.67	47.17
12 /40	0.500	98.191	0.95089	80.638	0.00100	0.00100	val_loss	3.23	47.27

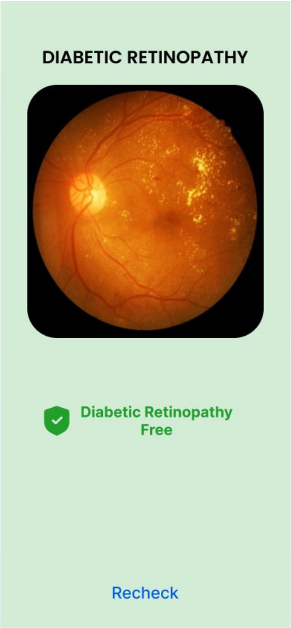
Result Comparison



User Interfaces







Work Continuation

Develop and train the model ✓

Implement Neural Network Search Algorithm

System Integration

IT20165666 | Lakshith G. P. R

Grade Severity of Diabetic Retinopathy using Retinal Fundus Images

Introduction

“To ensure effective treatments for DR it is crucial to determine the patient’s current stage of DR.”

- National Eye Institute -

Introduction

- 1/3 of Sri Lankan adults with self-reported diabetes have retinopathy [2]
- 40 board-certified ophthalmologists and 6 vitreo-retinal surgeons in the region [1]
- 77.5% (31 out of 40) specialists in Colombo district [1]
- Highest DR infrastructure ratios in Colombo [1]
- Medical officers' DR screening skills are low [1]
- Western province lacks systematic DR screening program [1]

Problem statement

How **effectively** leverage technological advancements in **machine learning and computer vision** for the early detection and **stage classification of Diabetic Retinopathy (DR)** from **retinal fundus images,** considering the **scarcity of human resources and expertise in the world** and the **lack of verified applications** in major digital stores?

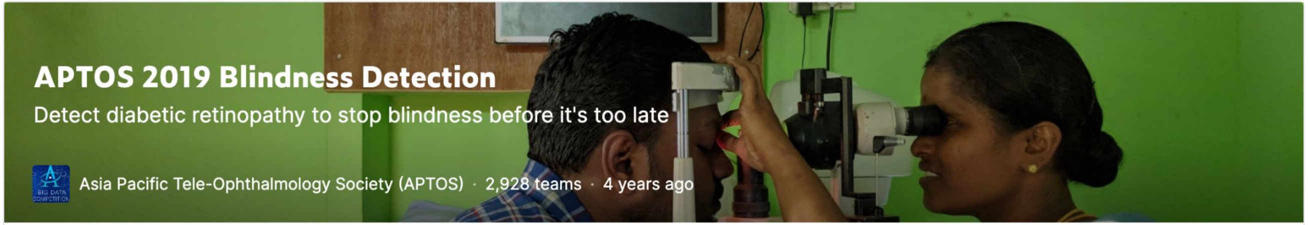
Main Objectives

- Develop a light weight deep learning- based algorithm
- Develop resource-efficient mobile app component

Sub Objectives

- Minimize the training time of the model
- Develop the model using minimum number of layers.
- Implement data augmentation techniques to balance the dataset
- Achieve more than the minimum test accuracy of 80%
- Deploy the model and the backend to a highly available cloud infrastructure

Dataset



Overview Data Code Discussion Leaderboard Rules Team ...

Dataset Description

You are provided with a large set of retina images taken using fundus photography under a variety of imaging conditions.

A clinician has rated each image for the severity of diabetic retinopathy on a scale of 0 to 4:

0 - No DR
1 - Mild
2 - Moderate
3 - Severe
4 - Proliferative DR

Like any real-world data set, you will encounter noise in both the images and labels. Images may contain artifacts, be out of focus, underexposed, or overexposed. The images were gathered from multiple clinics using a variety of cameras over an extended period of time, which will introduce further variation.

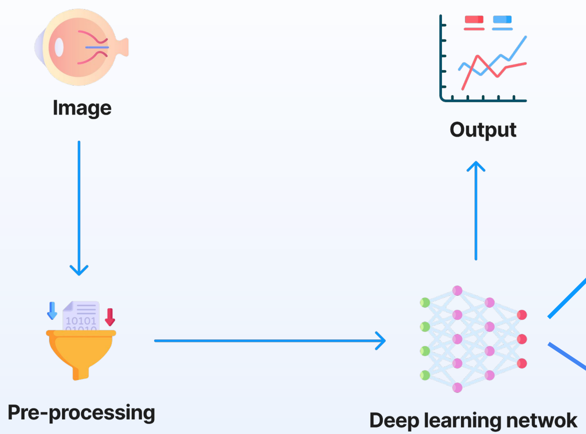
Files
5593 files

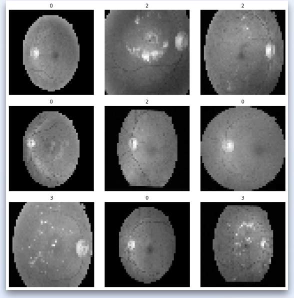
Size
10.22 GB

Type
png, csv

Visit for dataset

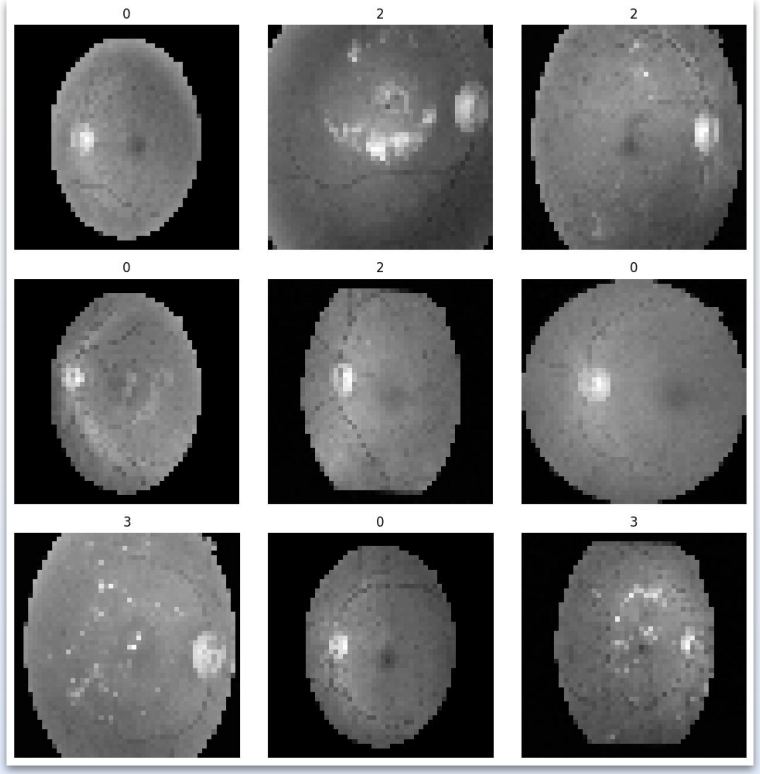
Component Implementation



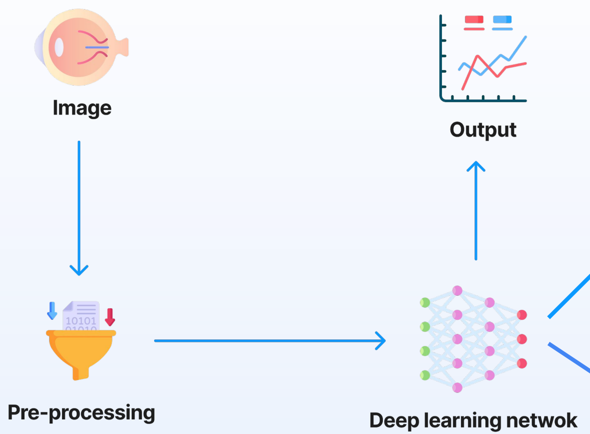


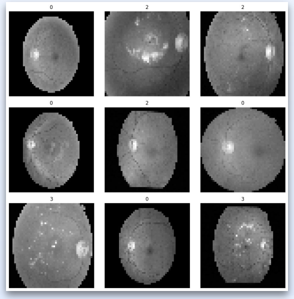
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 64)        640

 max_pooling2d (MaxPooling2D  (None, 24, 24, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 12, 12, 64)       0
 2D)

 dropout (Dropout)           (None, 12, 12, 64)        0

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 5)                 46085

=================================================================
Total params: 83,653
Trainable params: 83,653
Non-trainable params: 0
_________________________________________________________________
```

Component Implementation



Component Implementation



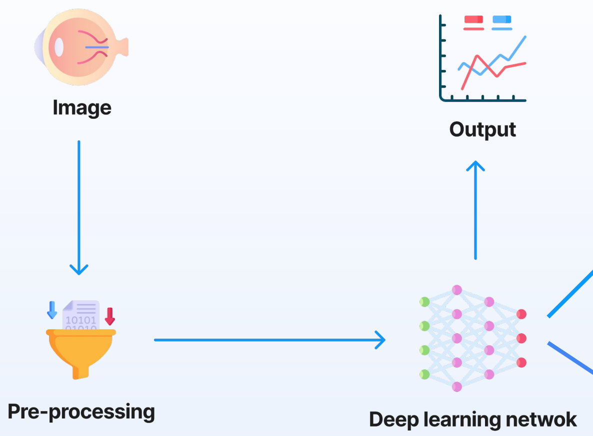


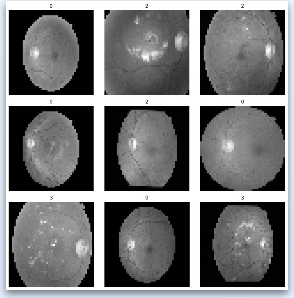
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 64)        640

 max_pooling2d (MaxPooling2D  (None, 24, 24, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 12, 12, 64)       0
 2D)

 dropout (Dropout)           (None, 12, 12, 64)        0

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 5)                 46085

=================================================================
Total params: 83,653
Trainable params: 83,653
Non-trainable params: 0
_________________________________________________________________
```

Component Implementation

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 64)        640

 max_pooling2d (MaxPooling2D  (None, 24, 24, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 12, 12, 64)       0
 2D)

 dropout (Dropout)           (None, 12, 12, 64)        0

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 5)                 46085

=================================================================
Total params: 83,653
Trainable params: 83,653
Non-trainable params: 0
_________________________________________________________________
```

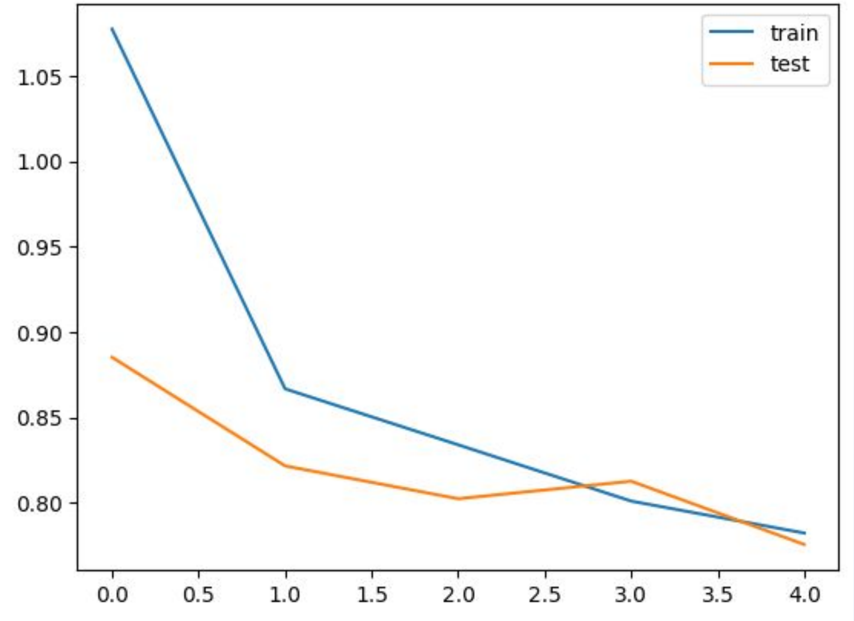
Component Implementation



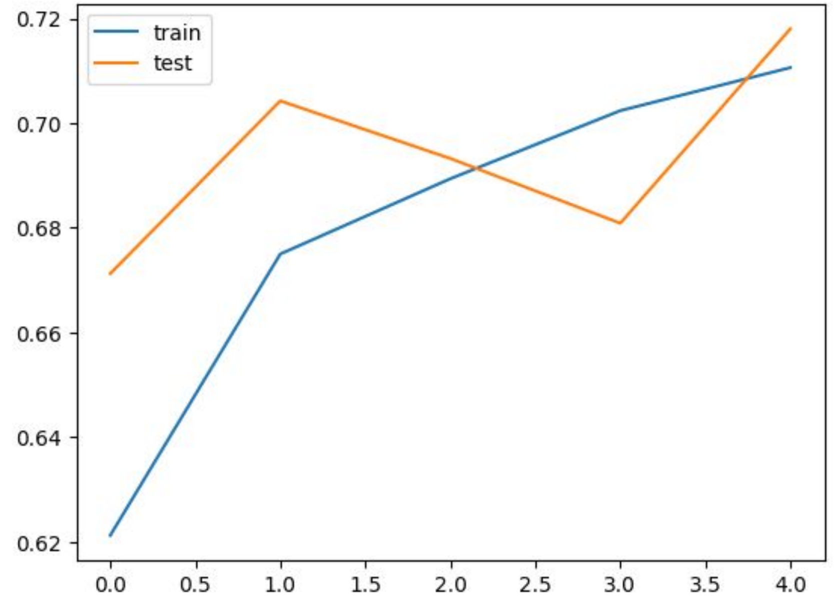


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 64)        640

 max_pooling2d (MaxPooling2D  (None, 24, 24, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 12, 12, 64)       0
 2D)

 dropout (Dropout)           (None, 12, 12, 64)        0

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 5)                 46085

=================================================================
Total params: 83,653
Trainable params: 83,653
Non-trainable params: 0
_________________________________________________________________
```

Results Comparison

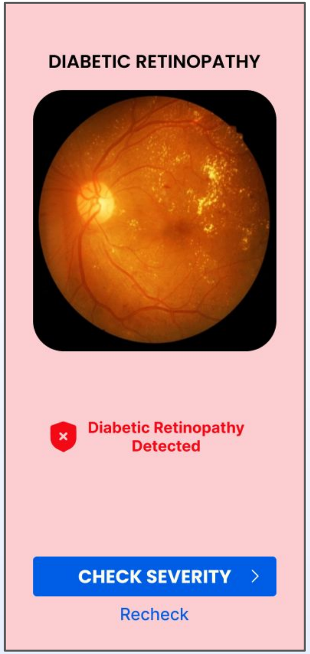


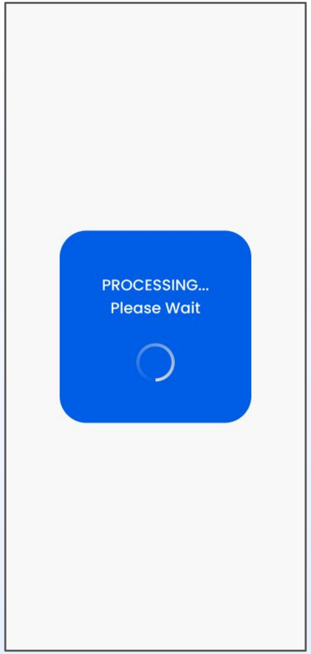
Loss
variation

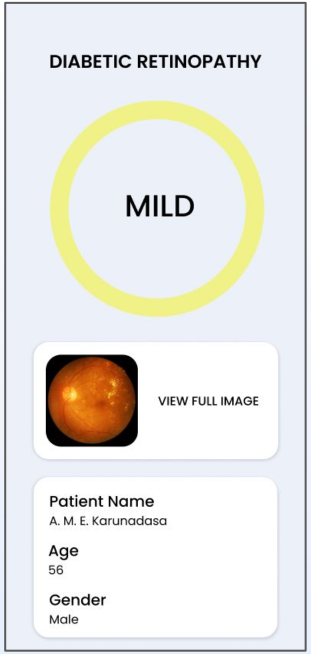


Accuracy
variation

User Interfaces







Work Continuation


Develop a deep learning based algorithm to accurately grade the severity of diabetic retinopathy in retinal images.
Background study
Topic evaluation
Literature review
Identify research gap
Identify research problem
feasibility study
Requirement gathering and analysing
Functional requirement gathering
Non-functional requirement gathering
Evaluate the requirements
Requirement specification
Design and implementation
Preparing the dataset
Design and implement pre-processing model
Design and implement retinal image classification model based on DR severity
Train the model
Design and ensemble model based on the above models
WORK IN PROGRESS
Client application development
WORK IN PROGRESS
Connect APIs and integrate with client application
WORK IN PROGRESS
Testing
Unit testing the models
Integration testing
System testing
Documentation
Topic assessment form
Project cover sheet
Project charter
Project proposal
Research paper
Final report
Student logbook
Project website

References

1. 1. M. M. P. N. Piyseana, G. V. S. Murthy, Availability of eye care infrastructure and human resources for managing diabetic retinopathy in the western province of Sri Lanka. Indian Journal of Ophthalmology. 68, 841–846 (2020).

2. Katulanda, P., Ranasinghe, P. & Jayawardena, R. Prevalence of retinopathy among adults with self-reported diabetes mellitus: the Sri Lanka diabetes and Cardiovascular Study. BMC Ophthalmol 14, 100 (2014). https://doi.org/10.1186/1471-2415-14-100

3. F. Alzami, Abdussalam, R. A. Megantara, A. Z. Fanani, and Purwanto, “Diabetic retinopathy grade classification based on fractal analysis and Random Forest,” 2019 International Seminar on Application for Technology of Information and Communication (iSemantic), 2019.

4. N. B. Thota and D. Umma Reddy, “Improving the accuracy of diabetic retinopathy severity classification with transfer learning,” 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), 2020.

5. Rajput, G.G.; Reshmi, B.; Rajesh, I. Automatic detection and grading of diabetic maculopathy using fundus images. Procedia Comput. Sci. 2020, 167, 57–66

6. Ahn, S.; Pham, Q.T.; Shin, J.; Song, S.J. Future Image Synthesis for Diab etic Retinopathy Based on the Lesion Occurrence Probability. Electronics 2021, 10, 726.

IT20227890 | Muthukumarana M. W. A. N. C.

Detect Symptoms of Age-related Macular Degeneration using Retinal OCT Images

01. Introduction

- Age-Related Macular Degeneration, is a progressive eye disease that affects the macula, the central part of the retina. It is a leading cause of vision loss among older adults.
- Early detection of AMD is crucial for effective treatment and prevention of further vision loss.

02. Research Problem

- Current process for AMD detection from OCT images is time-consuming and reliant on human observers.
- Misdiagnosis or delayed diagnosis of AMD can result in irreversible vision loss and decreased quality of life.
- Lack of automated and accurate diagnostic tools poses a significant challenge to early detection and treatment of AMD.

02. Main Objectives

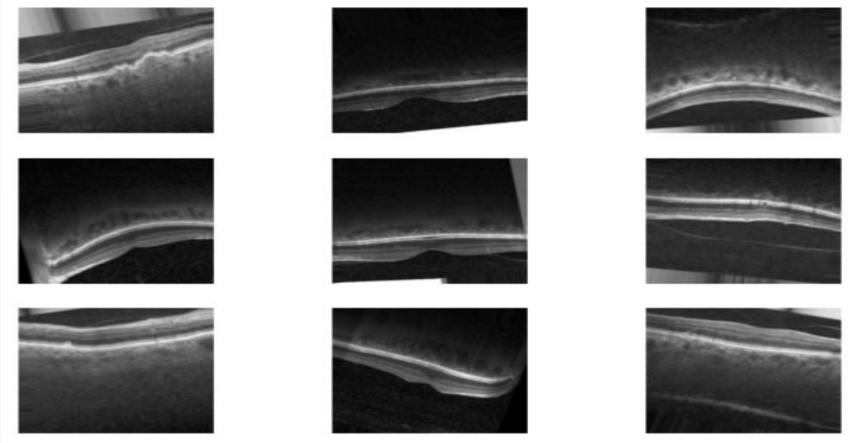
- Develop a machine learning model for the detection of age-related macular degeneration (AMD) from optical coherence tomography (OCT) images.

03. Sub Objectives

- Dataset collection
- Image preprocessing
- Deep learning model development
- Model optimization

04. Datasets

- There are Total Samples 15,900 of OCT images
    - 7950 AMD OCT images
    - 7950 Normal OCT Images
- https://www.kaggle.com/datasets/obulisainaren/retinal-oct-c8



05. Implementation of Component

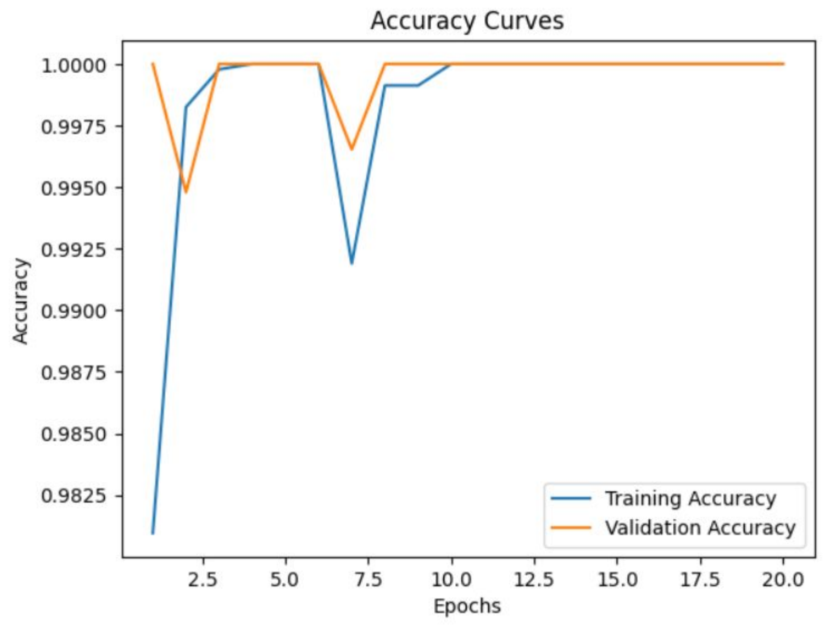
- Frameworks and Libraries
  - Tensorflow
  - keras
- Model Architecture Summary
  - Conv2D
  - MaxPooling2D
  - Flatten
  - Dense
  - Dropout
- Loss Function and Optimization
  - Adam

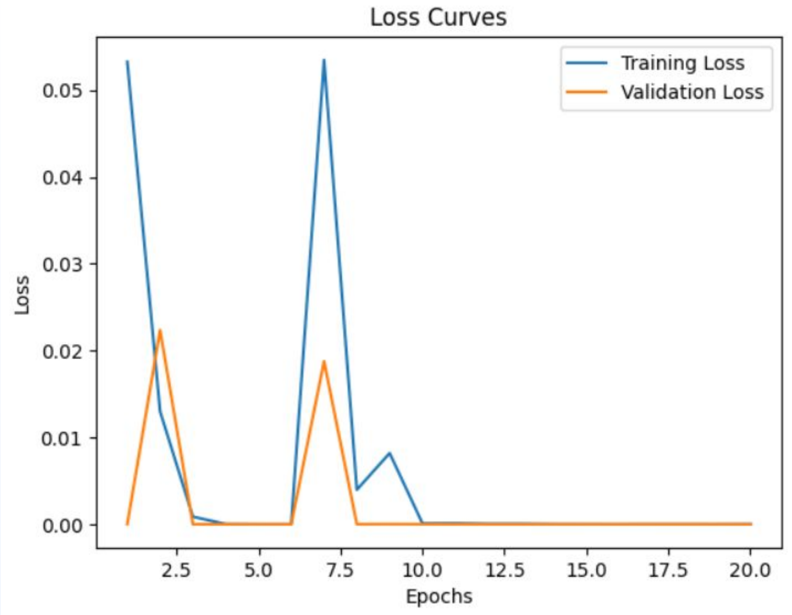
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1319 (Conv2D)        (None, 222, 222, 32)      320

 max_pooling2d_59 (MaxPoolin  (None, 111, 111, 32)     0
 g2D)

 conv2d_1320 (Conv2D)        (None, 109, 109, 64)      18496

 max_pooling2d_60 (MaxPoolin  (None, 54, 54, 64)       0
 g2D)

 conv2d_1321 (Conv2D)        (None, 52, 52, 128)       73856

 max_pooling2d_61 (MaxPoolin  (None, 26, 26, 128)      0
 g2D)

 flatten_2 (Flatten)         (None, 86528)             0

 dense_21 (Dense)            (None, 128)               11075712

 dropout_9 (Dropout)         (None, 128)               0

 dense_22 (Dense)            (None, 1)                 129

=================================================================
Total params: 11,168,513
Trainable params: 11,168,513
Non-trainable params: 0
```

06. Result Comparison

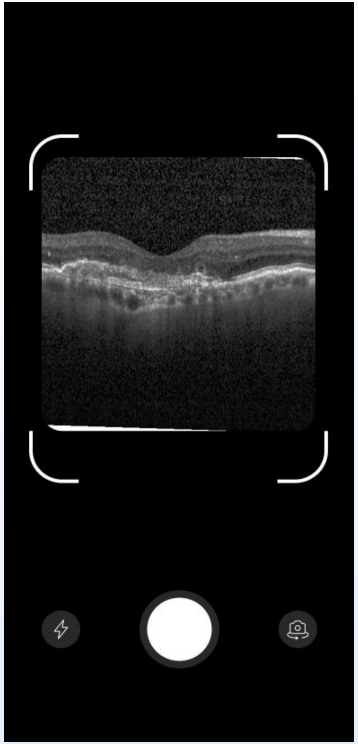
```
Epoch 1/20
143/143 [==============================] - 25s 164ms/step - loss: 0.0533 - accuracy: 0.9810 - val_loss: 8.8125e-06 - val_accuracy: 1.0000
Epoch 2/20
143/143 [==============================] - 25s 175ms/step - loss: 0.0130 - accuracy: 0.9982 - val_loss: 0.0224 - val_accuracy: 0.9948
Epoch 3/20
143/143 [==============================] - 22s 151ms/step - loss: 8.6288e-04 - accuracy: 0.9998 - val_loss: 6.4991e-08 - val_accuracy: 1.0000
Epoch 4/20
143/143 [==============================] - 26s 185ms/step - loss: 2.1652e-05 - accuracy: 1.0000 - val_loss: 1.1854e-08 - val_accuracy: 1.0000
Epoch 5/20
143/143 [==============================] - 22s 155ms/step - loss: 4.1566e-06 - accuracy: 1.0000 - val_loss: 5.7801e-09 - val_accuracy: 1.0000
Epoch 6/20
143/143 [==============================] - 25s 174ms/step - loss: 2.2885e-06 - accuracy: 1.0000 - val_loss: 4.2507e-09 - val_accuracy: 1.0000
Epoch 7/20
143/143 [==============================] - 22s 151ms/step - loss: 0.0535 - accuracy: 0.9919 - val_loss: 0.0188 - val_accuracy: 0.9965
Epoch 8/20
143/143 [==============================] - 25s 172ms/step - loss: 0.0040 - accuracy: 0.9991 - val_loss: 4.1885e-06 - val_accuracy: 1.0000
Epoch 9/20
143/143 [==============================] - 25s 176ms/step - loss: 0.0082 - accuracy: 0.9991 - val_loss: 1.3567e-05 - val_accuracy: 1.0000
Epoch 10/20
143/143 [==============================] - 23s 161ms/step - loss: 9.2370e-05 - accuracy: 1.0000 - val_loss: 2.8335e-06 - val_accuracy: 1.0000
Epoch 11/20
143/143 [==============================] - 27s 186ms/step - loss: 1.0697e-04 - accuracy: 1.0000 - val_loss: 2.4121e-06 - val_accuracy: 1.0000
Epoch 12/20
143/143 [==============================] - 25s 173ms/step - loss: 5.1908e-05 - accuracy: 1.0000 - val_loss: 8.3272e-07 - val_accuracy: 1.0000
```

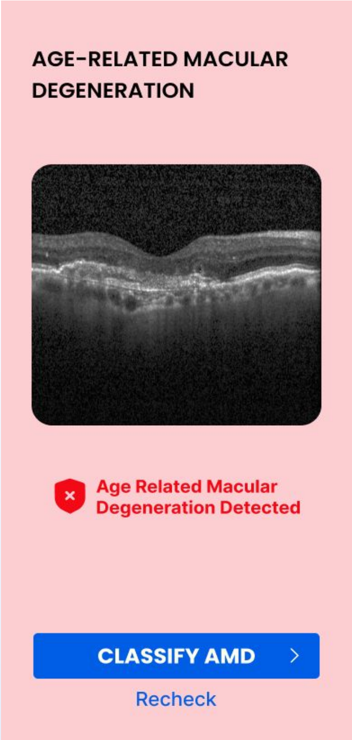
06. Result Comparison

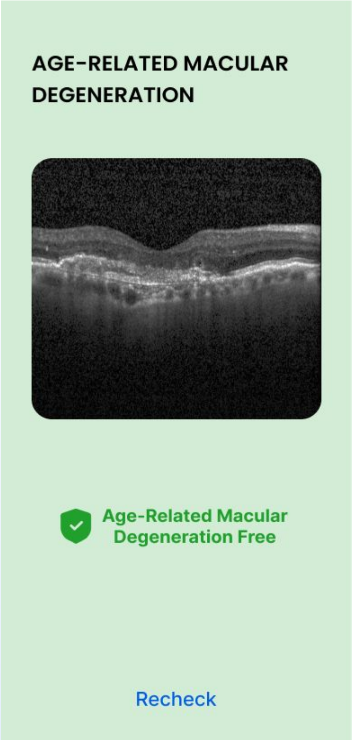




07. User Interfaces







IT20227890 | Muthukumarana M.W.A.N.C | 2023-162

Work Continuation

Develop and train the model

Optimizing the model for mobile devices

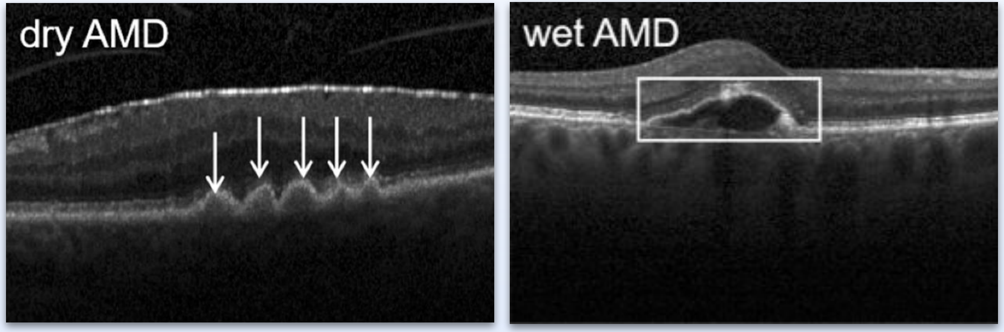
System Integration

IT20172978 | Kariyawasam K. G. P. C.

Classification of Age-related Macular Degeneration using Retinal OCT Images

Introduction

- Age-related macular degeneration(AMD) is a significant health burden that can lead to irreversible vision loss in the elderly.
- There are two types of AMD:
    - Dry AMD (Drusen)
    - Wet AMD (CNV)



https://www.macular.org/about-macular-degeneration/what-is-macular-degeneration/types

Research Problem

- Accurate classification of dry and wet AMD from a single OCT image.
- Challenges of time and cost in AMD diagnosis.
- Ability to classify AMD without any assistance of an opthamologist.
- Being able to optimize outcomes for a mobile application.

Main Objective

- Develop a novel DL model to accurately detect and classify wet and dry AMD based on a single OCT image while providing annotations for the affected regions.

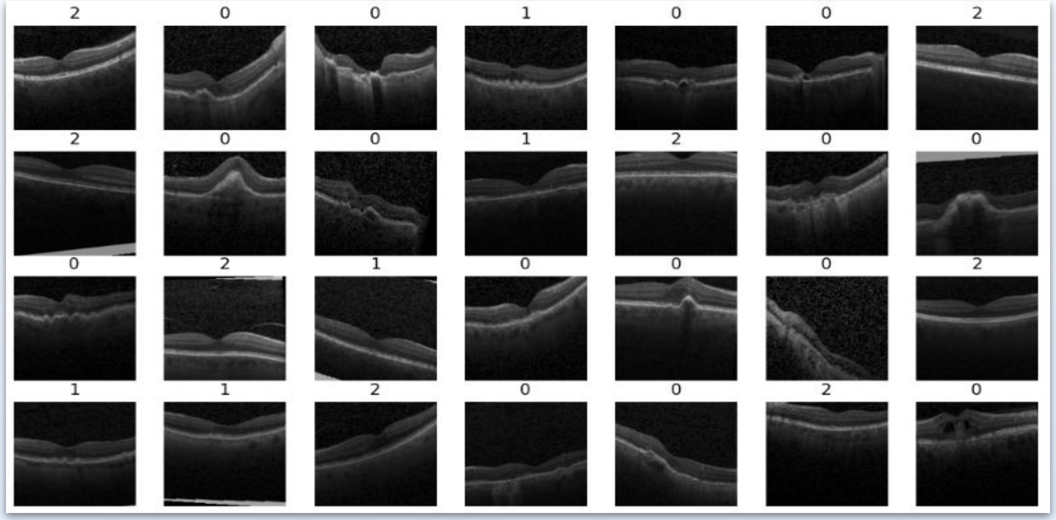
Sub Objectives

- Data Collection and Preprocessing
- Develop the deep learning model
- Evaluate and validate the model’s performance
- Annotation Generation for Affected Regions

Datasets

- Drusen - 8600 OCT Images
- CNV - 13000 OCT Images
- Normal - 20000 OCT Images

https://www.kaggle.com/datasets/paultimothymooney/kermany2018



Implementation of component

- Frameworks and Libraries
    - Tensorflow
    - keras
- Model Architecture Summary
    - Conv2D
    - MaxPooling2D
    - Dense
    - Dropout
- Loss Function and Optimization
    - Categorical Cross-entropy
    - Adam

Model: "model"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 150, 150, 3)]	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 256)	2097408
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 3)	771

Model Implementation

AMD_Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Preprocessing

```python
#Calculate class weight for the imbalanced data
total = 72136

count_cnv = 37205
count_drusen = 8616
count_normal = 26315

cnv_weight = (1/count_cnv) * (total/3)
drusen_weight = (1/count_drusen) * (total/3)
norm_weight = (1/count_normal) * (total/3)

class_weight = {0 : cnv_weight, 1 : drusen_weight, 2: norm_weight}
print(class_weight)


labels = ['CNV', 'DRUSEN', 'NORMAL']

def load(path):
    listed_paths = []
    for i in labels:
        base_path = os.path.join(path, i)
        imgs = os.listdir(base_path)
        for img in imgs:
            img_path = base_path + '/' + img
            listed_paths.append(img_path)
    return listed_paths

train_paths = load(train_path)
val_paths = load(val_path)
test_paths = load(test_path)
```

```
{0: 0.6462930609685078, 1: 2.790776849272671, 2: 0.9137500791690417}
```

AMD_Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Build the Model VGG16

```python
# pre-trained VGG16 model

base_model = VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
trainable_layers = len(base_model.layers[:-5])
for layer in base_model.layers[:trainable_layers]:
    layer.trainable = False # Freeze the weights of all layers except for the last 5
for layer in base_model.layers[trainable_layers:]:
    layer.trainable = True
```

```python
# Add new classification layers on top of the pre-trained layers
num_classes = 3  # cnv, drusen, and normal.
x = Flatten()(base_model.output)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x) # Add a dropout layer with a rate of 0.5
predictions = Dense(num_classes, activation='softmax')(x)
```

```python
# Create the final model
model_3 = Model(inputs=base_model.input, outputs=predictions)
# Compile the model with Adam optimizer
opt = Adam(lr=0.0001)
model_3.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
# Define the callbacks
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=0.00001)
early_stop = EarlyStopping(monitor='val_loss', patience=5)
# Train the model
num_epochs = 20
history_3 = model_3.fit(train_generator,
                        epochs = num_epochs,
                        steps_per_epoch = 500,
                        validation_data = val_generator,
                        validation_steps = len(val_generator),
                        callbacks = [reduce_lr, early_stop])
```

Result Comparison

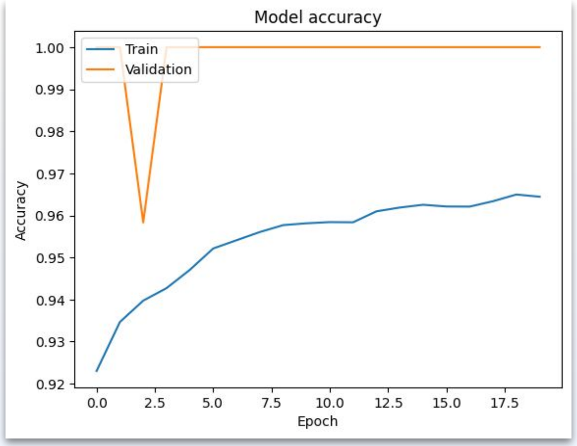
AMD_Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

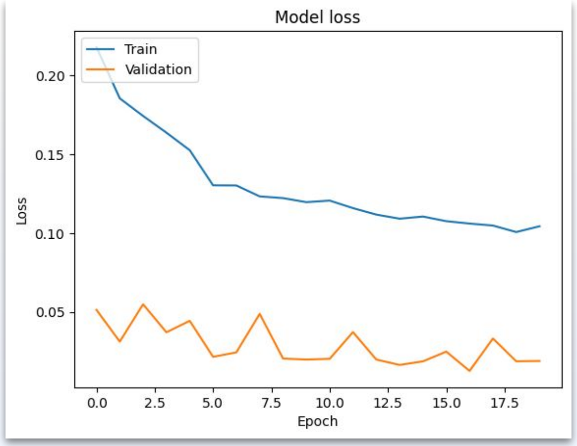
+ Code + Text

```
Epoch 1/20
500/500 [==============================] - 348s 695ms/step - loss: 0.2179 - accuracy: 0.9230 - val_loss: 0.0513 - val_accuracy: 1.0000 - lr: 1.0000e-04
Epoch 2/20
500/500 [==============================] - 341s 682ms/step - loss: 0.1854 - accuracy: 0.9347 - val_loss: 0.0311 - val_accuracy: 1.0000 - lr: 1.0000e-04
Epoch 3/20
500/500 [==============================] - 338s 676ms/step - loss: 0.1743 - accuracy: 0.9397 - val_loss: 0.0548 - val_accuracy: 0.9583 - lr: 1.0000e-04
Epoch 4/20
500/500 [==============================] - 337s 675ms/step - loss: 0.1637 - accuracy: 0.9427 - val_loss: 0.0370 - val_accuracy: 1.0000 - lr: 1.0000e-04
Epoch 5/20
500/500 [==============================] - 328s 656ms/step - loss: 0.1526 - accuracy: 0.9471 - val_loss: 0.0443 - val_accuracy: 1.0000 - lr: 1.0000e-04
Epoch 6/20
500/500 [==============================] - 326s 652ms/step - loss: 0.1303 - accuracy: 0.9521 - val_loss: 0.0215 - val_accuracy: 1.0000 - lr: 2.0000e-05
Epoch 7/20
500/500 [==============================] - 326s 652ms/step - loss: 0.1302 - accuracy: 0.9541 - val_loss: 0.0243 - val_accuracy: 1.0000 - lr: 2.0000e-05
Epoch 8/20
500/500 [==============================] - 325s 649ms/step - loss: 0.1233 - accuracy: 0.9560 - val_loss: 0.0487 - val_accuracy: 1.0000 - lr: 2.0000e-05
Epoch 9/20
500/500 [==============================] - 324s 648ms/step - loss: 0.1222 - accuracy: 0.9577 - val_loss: 0.0204 - val_accuracy: 1.0000 - lr: 2.0000e-05
Epoch 10/20
500/500 [==============================] - 323s 646ms/step - loss: 0.1196 - accuracy: 0.9581 - val_loss: 0.0197 - val_accuracy: 1.0000 - lr: 2.0000e-05
Epoch 11/20
500/500 [==============================] - 325s 650ms/step - loss: 0.1206 - accuracy: 0.9584 - val_loss: 0.0202 - val_accuracy: 1.0000 - lr: 2.0000e-05
Epoch 12/20
500/500 [==============================] - 323s 645ms/step - loss: 0.1158 - accuracy: 0.9584 - val_loss: 0.0372 - val_accuracy: 1.0000 - lr: 2.0000e-05
Epoch 13/20
500/500 [==============================] - 364s 727ms/step - loss: 0.1117 - accuracy: 0.9610 - val_loss: 0.0197 - val_accuracy: 1.0000 - lr: 2.0000e-05
Epoch 14/20
500/500 [==============================] - 330s 659ms/step - loss: 0.1091 - accuracy: 0.9619 - val_loss: 0.0163 - val_accuracy: 1.0000 - lr: 1.0000e-05
Epoch 15/20
500/500 [==============================] - 317s 634ms/step - loss: 0.1105 - accuracy: 0.9625 - val_loss: 0.0186 - val_accuracy: 1.0000 - lr: 1.0000e-05
Epoch 16/20
500/500 [==============================] - 313s 626ms/step - loss: 0.1075 - accuracy: 0.9621 - val_loss: 0.0248 - val_accuracy: 1.0000 - lr: 1.0000e-05
Epoch 17/20
500/500 [==============================] - 316s 632ms/step - loss: 0.1060 - accuracy: 0.9621 - val_loss: 0.0125 - val_accuracy: 1.0000 - lr: 1.0000e-05
Epoch 18/20
500/500 [==============================] - 315s 630ms/step - loss: 0.1048 - accuracy: 0.9634 - val_loss: 0.0331 - val_accuracy: 1.0000 - lr: 1.0000e-05
Epoch 19/20
500/500 [==============================] - 314s 628ms/step - loss: 0.1006 - accuracy: 0.9650 - val_loss: 0.0187 - val_accuracy: 1.0000 - lr: 1.0000e-05
Epoch 20/20
500/500 [==============================] - 314s 627ms/step - loss: 0.1043 - accuracy: 0.9644 - val_loss: 0.0188 - val_accuracy: 1.0000 - lr: 1.0000e-05
```

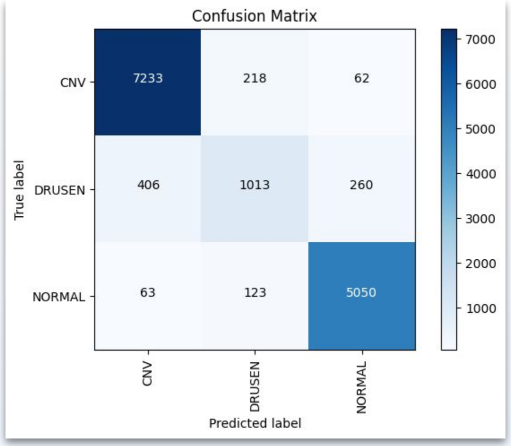
Evidence



Plot training & validation accuracy values



Plot training & validation loss values

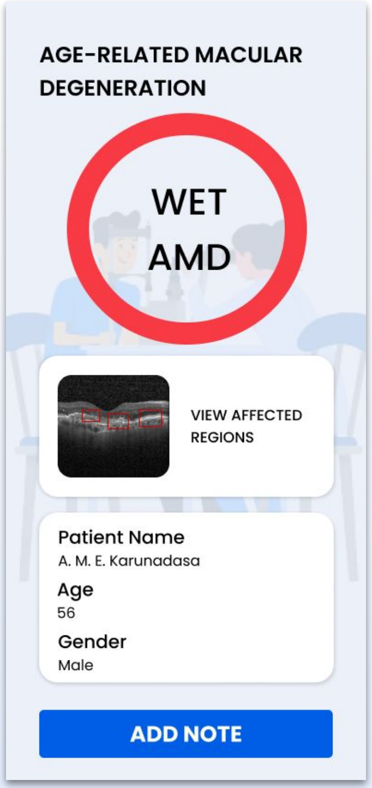


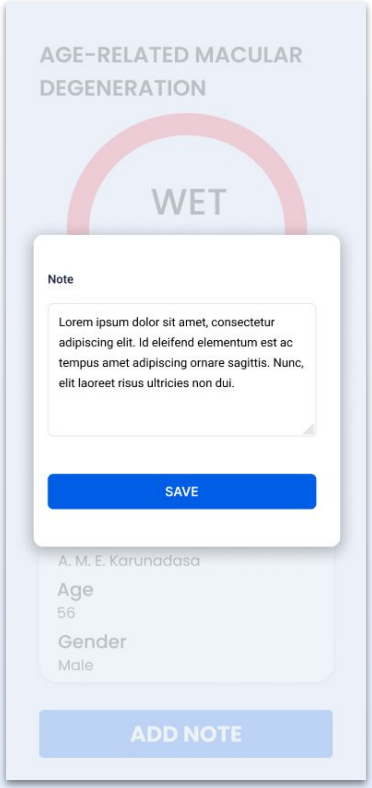
Confusion Matrix

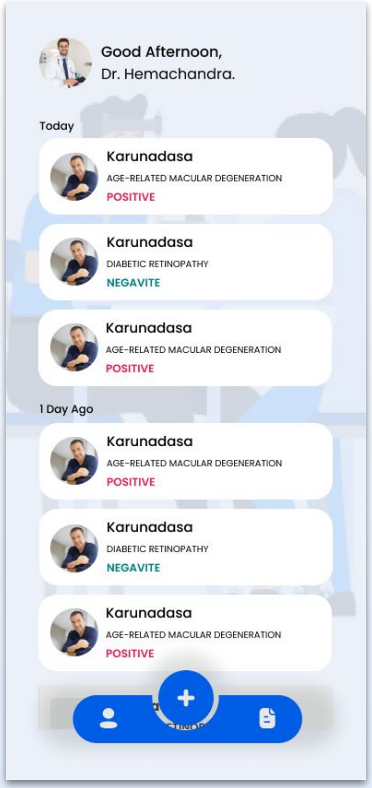
```
Train accuracy: 0.9542208164930344
Train loss: 0.13051368705928326

Validation accuracy: 1.0
Validation loss: 0.024401552975177765

Test accuracy: 0.9820936918258667
Test loss: 0.05077371001243591
```

Train, Validation, and Test Accuracies

User Interfaces







Work Continuation

Develop and train the model with the dataset for AMD classification

Accurately annotate the affected regions of Dry or Wet AMD OCT images and implement user interfaces

System Integration

Thank You !