



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

2023-162

Our Team



Team Leader
IT20166106



Member 01
IT20165666



Member 02
IT20227890



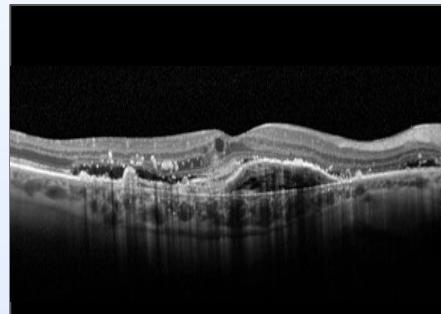
Member 03
IT20172978

Introduction

- Development of a mobile application for the identification of Diabetic Retinopathy and Age-Related Macular Degeneration to aid eye specialists with accurate diagnoses.



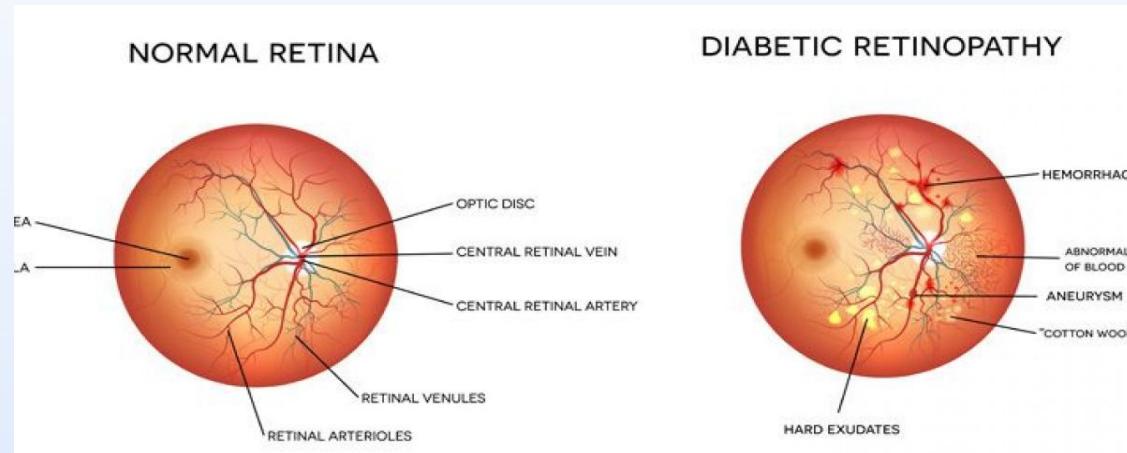
Fundus Image



OCT Image

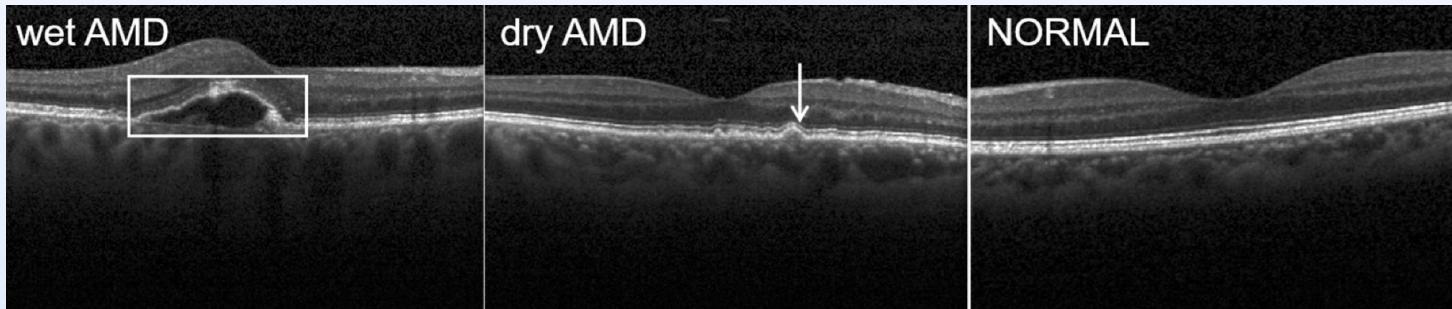
What is Diabetic Retinopathy

- 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

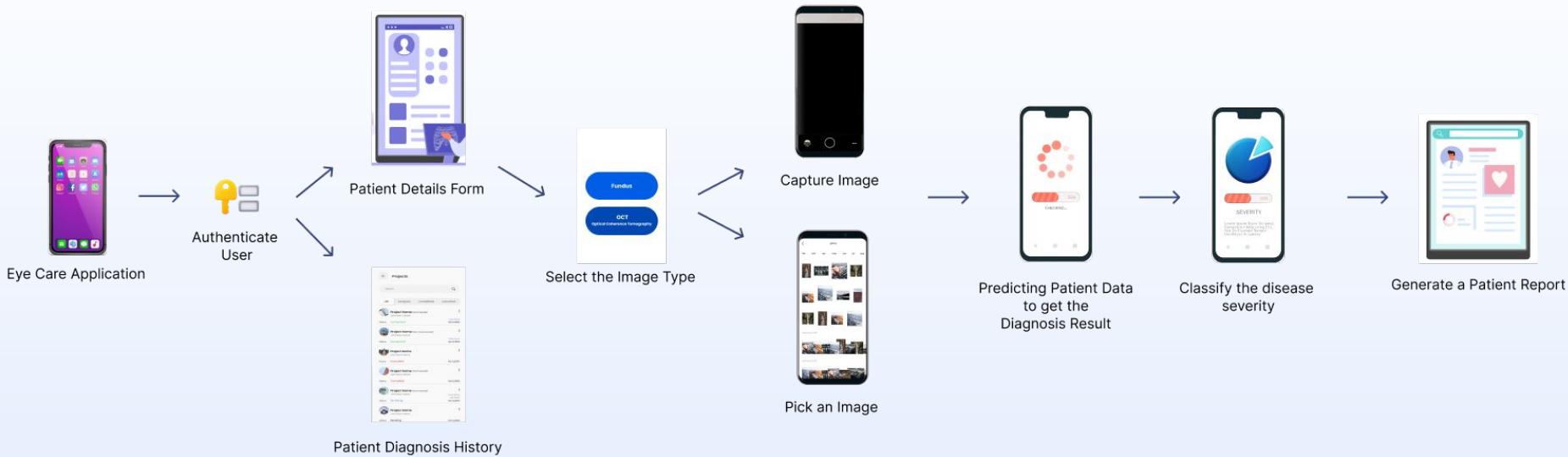
- Age-Related Macular Degeneration (AMD) causes progressive vision loss and affects millions worldwide.
- There are two types of AMD:
 - Dry AMD (Drusen)
 - Wet AMD (CNV)



Research Problem

- Unavailability of a verified mobile application to automate the diagnose process.
- Diagnosis of eye diseases is time-consuming and requires expertise and training.
- Accurate detection of eye diseases are crucial to avoid visual impairments

User Flow Diagram



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



IT20166106 | Perera H. A. N. S

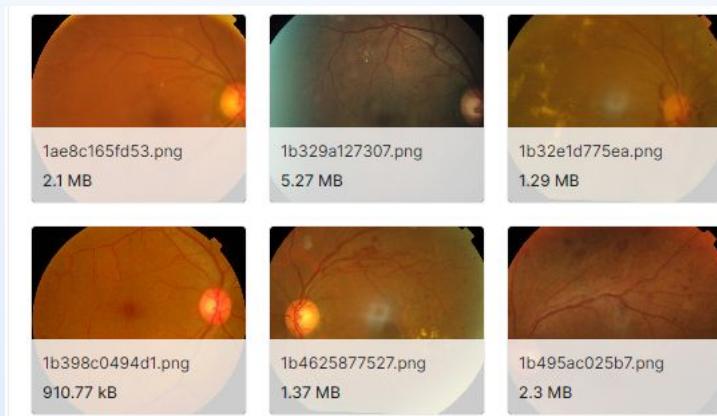
Detect Symptoms of Diabetic Retinopathy using Retinal Fundus Images

Introduction

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

Datasets

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



Implementation of Component

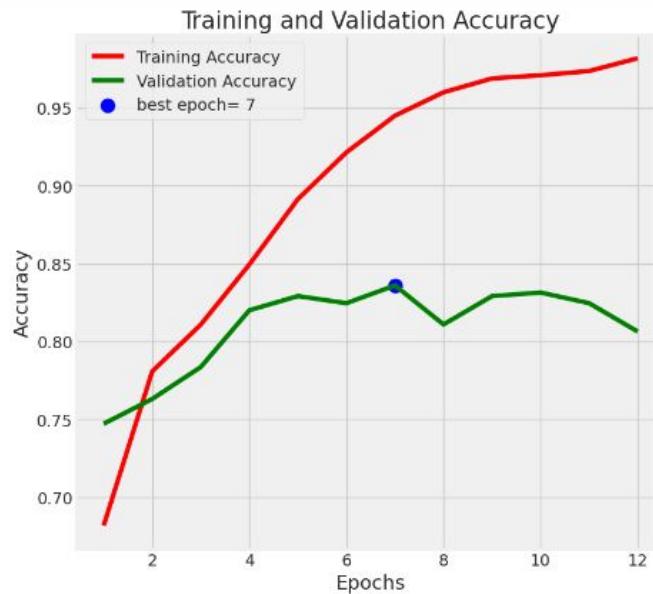
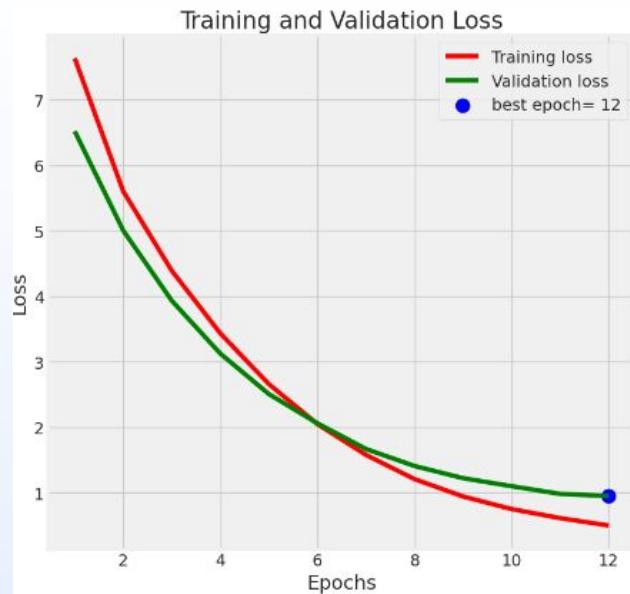
- Frameworks and Libraries
 - Tensorflow
 - keras
- Model Architecture Summary
 - EfficientNetB3
 - BatchNormalization
 - Dense
 - Dropout
- Loss Function and Optimization
 - Adam



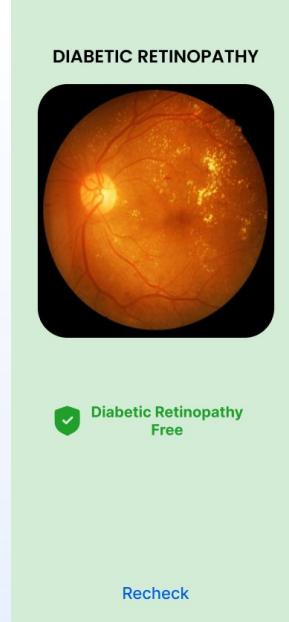
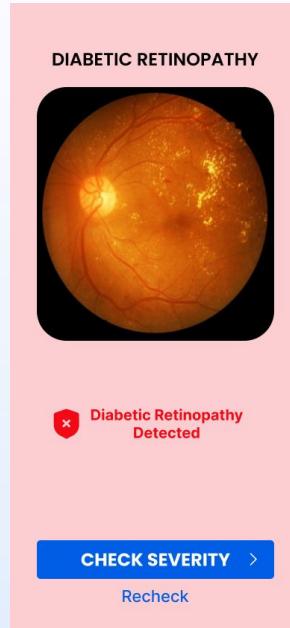
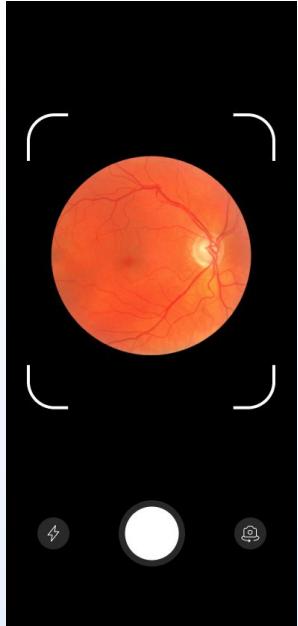
Result Comparison

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

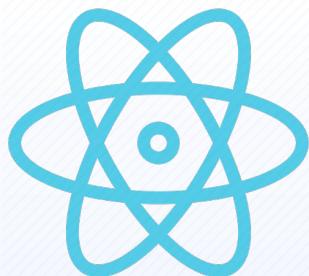
Result Comparison



User Interfaces



Frontend & Backend - Technologies



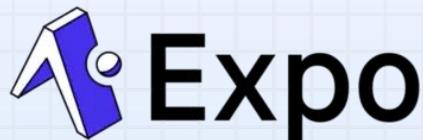
React Native



Flask

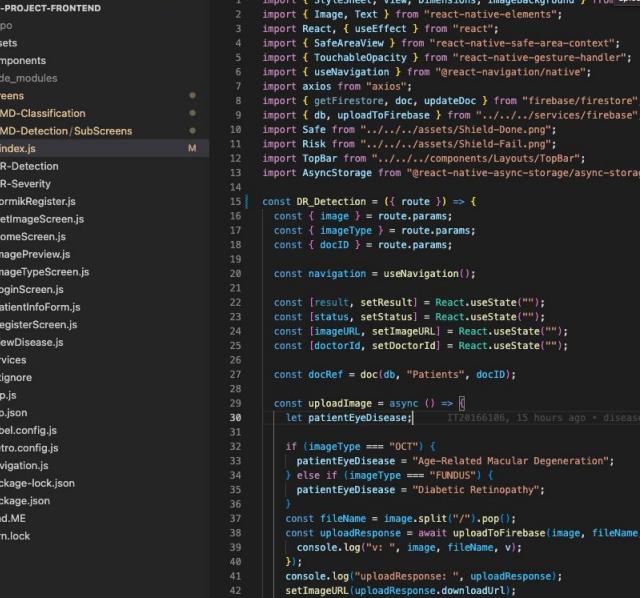


Firebase



FORMIK

Frontend & Backend - Code



The screenshot shows a Mac OS X desktop with a React Native project open in an IDE. The project structure is as follows:

- EXPLORER**: Shows files and folders:
 - OPEN EDITORS**: index.js (screens/AMD-Detection/SubScreens)
 - CDAP-PROJECT-FRONTEND**:
 - .expo
 - assets
 - components
 - node_modules
 - screens
 - AMD-Detection
 - GetImageScreen.js
 - HomeScreen.js
 - ImagePreview.js
 - ImageTypeScreen.js
 - LoginScreen.js
 - PatientInfoForm.js
 - RegisterScreen.js
 - viewDisease.js
 - services
 - .gitignore
 - App.js
 - app.json
 - babel.config.js
 - metro.config.js
 - Navigation.js
 - package-lock.json
 - package.json
 - read.ME
 - yarn.lock

OUTLINE: Shows the file tree.

JS index.js (M) **X**

index.js – cdap-project-frontend

```
index.js M < JS index.js M < JS index.js > AMD-Detection > SubScreens > JS index.js > DR_Detection > uploadImage
index.js - 1, You 1 second ago | 4 authors (IT20166106 and others)
1 import { StyleSheet, View, Dimensions, ImageBackground } from "react-native";
2 import { Image, Text } from "react-native-elements";
3 import React, { useEffect } from "react";
4 import { SafeAreaView } from "react-native-safe-area-context";
5 import { TouchableOpacity } from "react-native-gesture-handler";
6 import { useNavigation } from "@react-navigation/native";
7 import axios from "axios";
8 import { getFirestore, doc, updateDoc } from "firebase/firestore";
9 import { db, uploadToFirebase } from "../../../../services/firebase";
10 import Safe from "../../../../../../assets/Shield-Done.png";
11 import Risk from "../../../../../../assets/Shield-Fail.png";
12 import TopBar from "../../../../Components/Layouts/TopBar";
13 import AsyncStorage from "@react-native-async-storage/async-storage";
14
15 const DR_Detection = ({ route }) => {
16   const { image } = route.params;
17   const { imageType } = route.params;
18   const { docID } = route.params;
19
20   const navigation = useNavigation();
21
22   const [result, setResult] = React.useState("");
23   const [status, setStatus] = React.useState("");
24   const [imageURL, setImageURL] = React.useState("");
25   const [doctorId, setDoctorId] = React.useState("");
26
27   const docRef = doc(db, "Patients", docID);
28
29   const uploadImage = async () => {
30     let patientEyeDisease; // IT20166106, 15 hours ago + disease detection screen
31
32     if (imageType === "OCT") {
33       patientEyeDisease = "Age-Related Macular Degeneration";
34     } else if (imageType === "FUNDUS") {
35       patientEyeDisease = "Diabetic Retinopathy";
36     }
37
38     const fileName = image.split("/").pop();
39     const uploadResponse = await uploadToFirebase(image, fileName, v) => {
40       console.log("v: ", image, fileName, v);
41     };
42     console.log("UploadResponse: ", uploadResponse);
43     setImageURL(uploadResponse.downloadUrl);
44
45     const patientDataRef = {
46       patientEyeDisease: patientEyeDisease,
47       img_url: uploadResponse.downloadUrl,
48       doctorId: doctorId,
49       status: "Negative",
50     };
51
52     updateDoc(docRef, patientDataRef).then(() => {
53       setStatus("Success");
54     }).catch((err) => {
55       console.log("Error updating document: ", err);
56     });
57   };
58
59   return (
60     <SafeAreaView>
61       <ImageBackground source={image} style={styles.image}>
62         <View style={styles.overlay} />
63         <Text style={styles.title}>{"DR-Detection"}/>
64         <Text style={styles.subtitle}>{"Predicting Eye Disease"}/>
65         <Text style={styles.info}>{"Upload your eye image"}/>
66         <Image style={styles.icon} source={imageType === "OCT" ? Risk : Safe} />
67         <Text style={styles.result}>{"Result"}</Text>
68         <Text style={styles.status}>{"Status"}</Text>
69         <Text style={styles.upload}>{"Upload Image"}/>
70       </ImageBackground>
71     </SafeAreaView>
72   );
73 }
74
75 export default DR_Detection;
```

The screenshot shows a Python development environment with the following details:

- File Explorer:** Shows files in the current workspace, including `main.py`, `requirements.txt`, and various Docker-related files.
- Code Editor:** The main editor window displays the `main.py` file content. The code implements two routes: `/dr/detection` and `/md/detection`. Both routes handle POST requests by fetching an image URL from the request, preprocessing it, and then using a TensorFlow model to predict the result. GET requests return an error message if no image URL is provided.
- Terminal:** A small terminal window at the bottom shows the command `main.py - TestPython`.

```
main.py - TestPython

main.py 3, M
main.py > ...
28 @app.route('/dr/detection', methods=['POST', 'GET'])
29 def dr_detection():
30     if request.method == 'GET':
31         image_url = request.args.get('image_url')
32         if not image_url:
33             return jsonify({'error': 'No image URL found'})
34
35         img_array = preprocess_image(image_url)
36         if img_array is None:
37             return jsonify({'error': 'Failed to fetch the image from the URL'})
38
39         return jsonify(result=predict_dr_result(img_array))
40
41     elif request.method == 'POST':
42         data = request.get_json()
43         image_url = data.get('image_url')
44
45         if not image_url:
46             return jsonify({'error': 'No image URL found in the request body'})
47
48         img_array = preprocess_image(image_url)
49         if img_array is None:
50             return jsonify({'error': 'Failed to fetch the image from the URL'})
51
52         return jsonify(result=predict_dr_result(img_array))
53
54     else:
55         return jsonify({'error': 'Method not allowed'})
56
57 @app.route('/md/detection', methods=['POST'])
58 def and_detection():
59     try:
60         data = request.get_json()
61         image_url = data['image_url']
62
63         # Download the image from the URL
64         response = requests.get(image_url)
65
66         if not response.content:
67             raise ValueError("Empty image content")
68
69         # Load the image using TensorFlow's load_img
70         img = tf.keras.preprocessing.image.load_img(BytesIO(response.content), color_mode='grayscale', target_size=(224, 224))
71
72         # Convert the image to a numpy array
73         x = tf.keras.preprocessing.image.img_to_array(img)
74
75         # Reshape the array to match the input shape of the model
76         x = np.expand_dims(x, axis=0)
```



IT20165666 | Lakshith G. P. R

Grade Severity of Diabetic Retinopathy using Retinal Fundus Images

Introduction

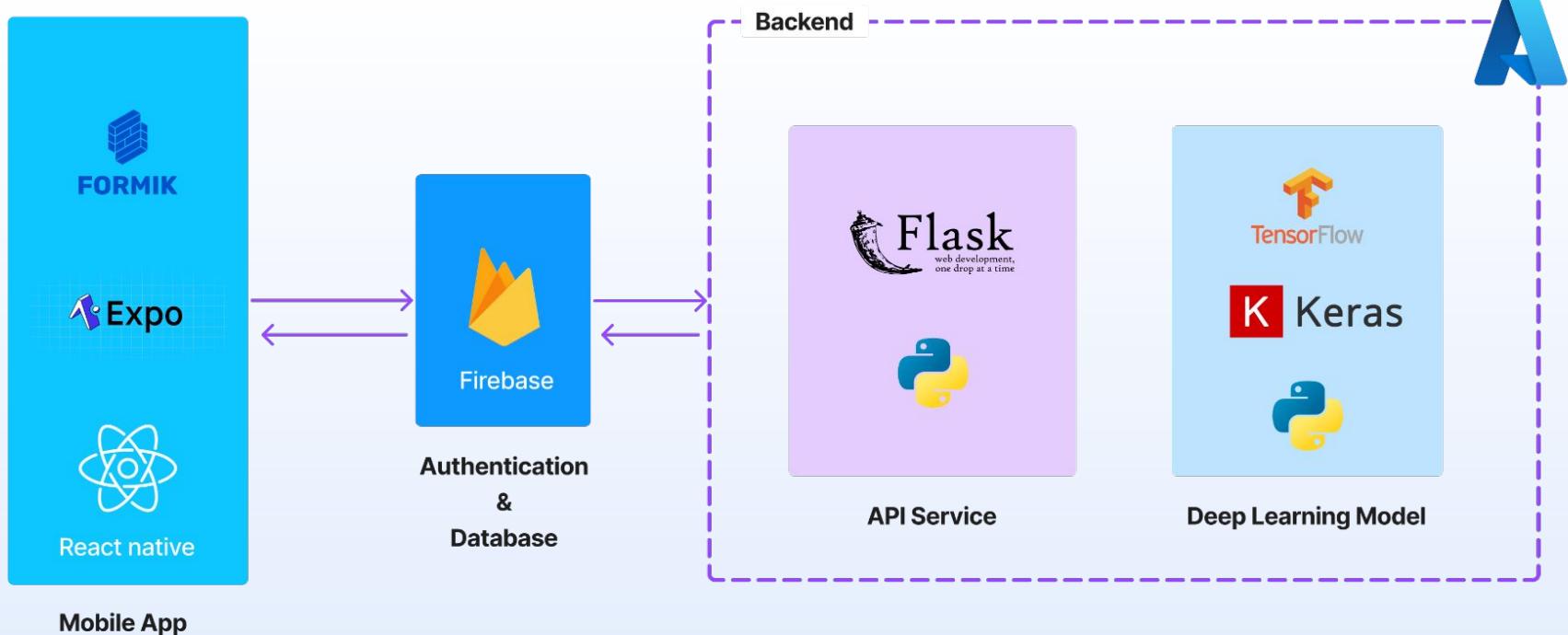
“To ensure effective treatments for DR it is crucial to determine the patient’s current stage of DR.”

- National Eye Institute -

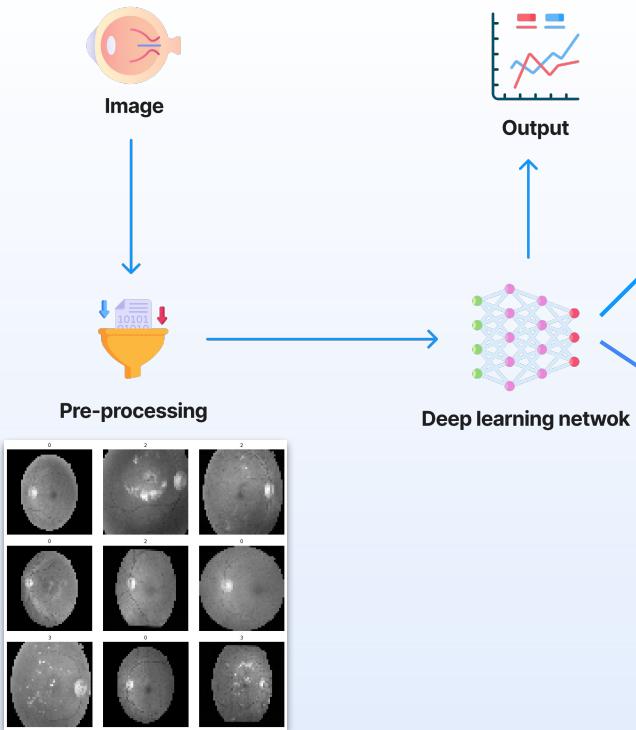
Objectives

- Develop a mobile optimized deep learning model
- Minimize the training time
- Deploy the model and backend into a highly available cloud infrastructure
- Develop a cross-platform support mobile app

Tools & Technologies

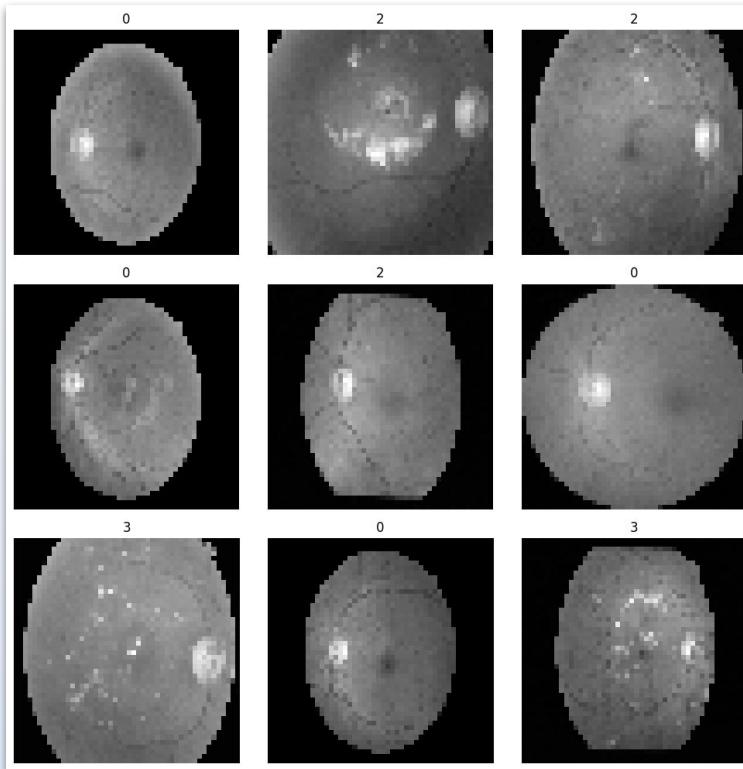


Component Implementation



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 (MaxPooling2D)	(None, 12, 12, 64)	0
dropout (Dropout)	(None, 12, 12, 64)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 5)	46085
<hr/>		
Total params: 83,653		
Trainable params: 83,653		
Non-trainable params: 0		

Component Imp. - Pre Processed Image



Component Imp. - Lightweight Model

```
Model: "sequential"
```

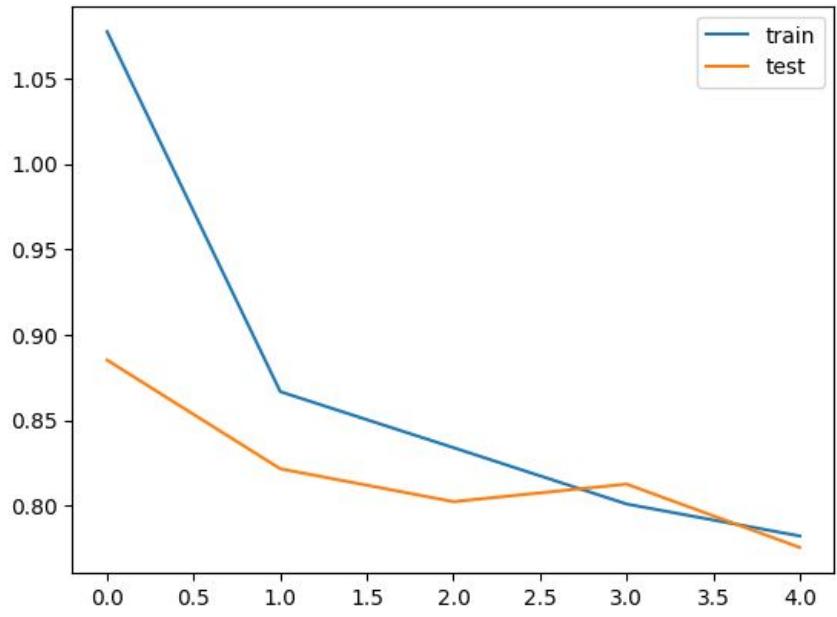
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
conv2d_1 (Conv2D)	(None, 24, 24, 64)	36928
max_pooling2d_1 (MaxPooling 2D)	(None, 12, 12, 64)	0
dropout (Dropout)	(None, 12, 12, 64)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 5)	46085

```
Total params: 83,653
```

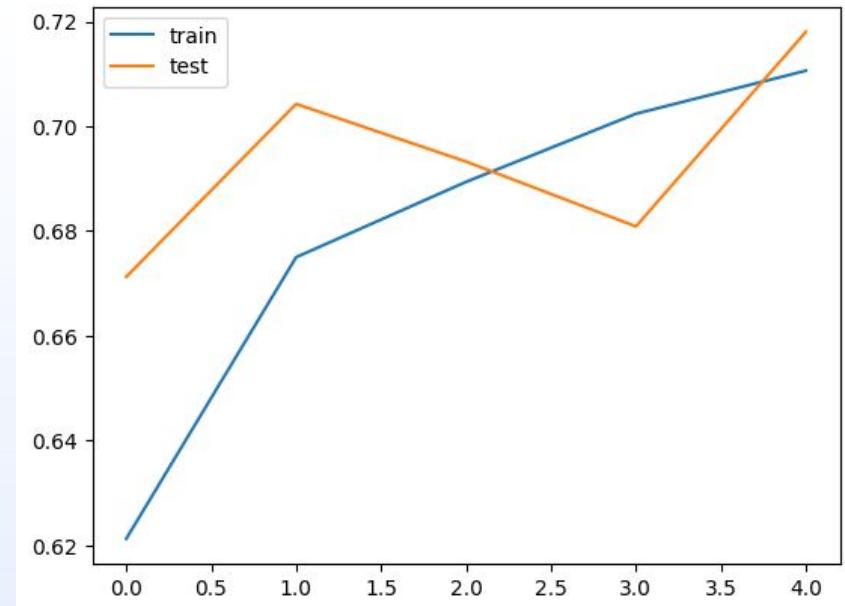
```
Trainable params: 83,653
```

```
Non-trainable params: 0
```

Result Comparison

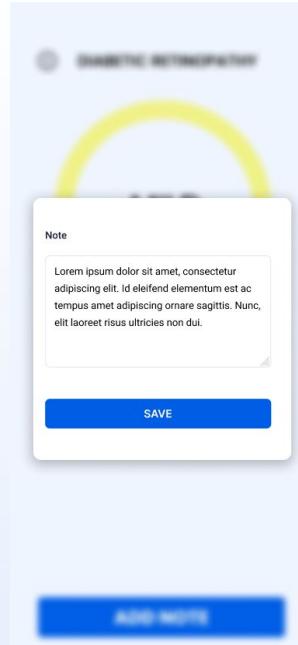
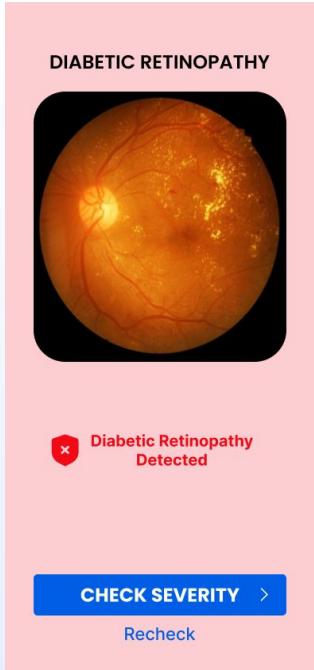


Loss
variation



Accuracy
variation

User Interface



Research Findings

- Data fragmentation techniques lead to model over-fitting
- Object identification algorithm to enhance results



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

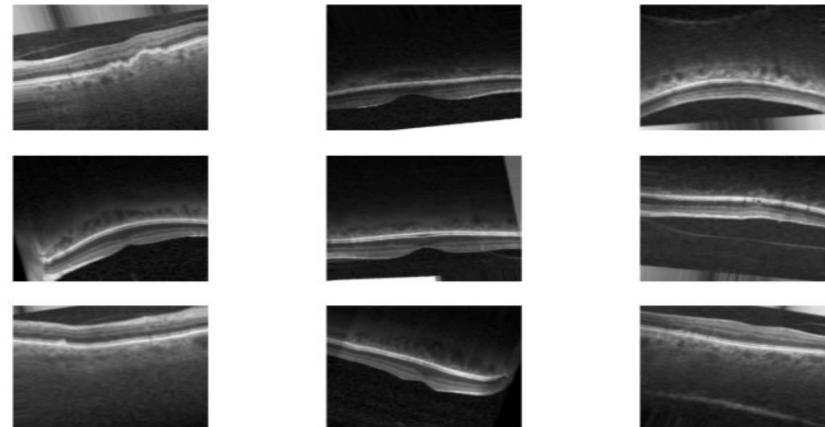
Detect Symptoms of Age-related Macular Degeneration using Retinal OCT Images

01. Introduction

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

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

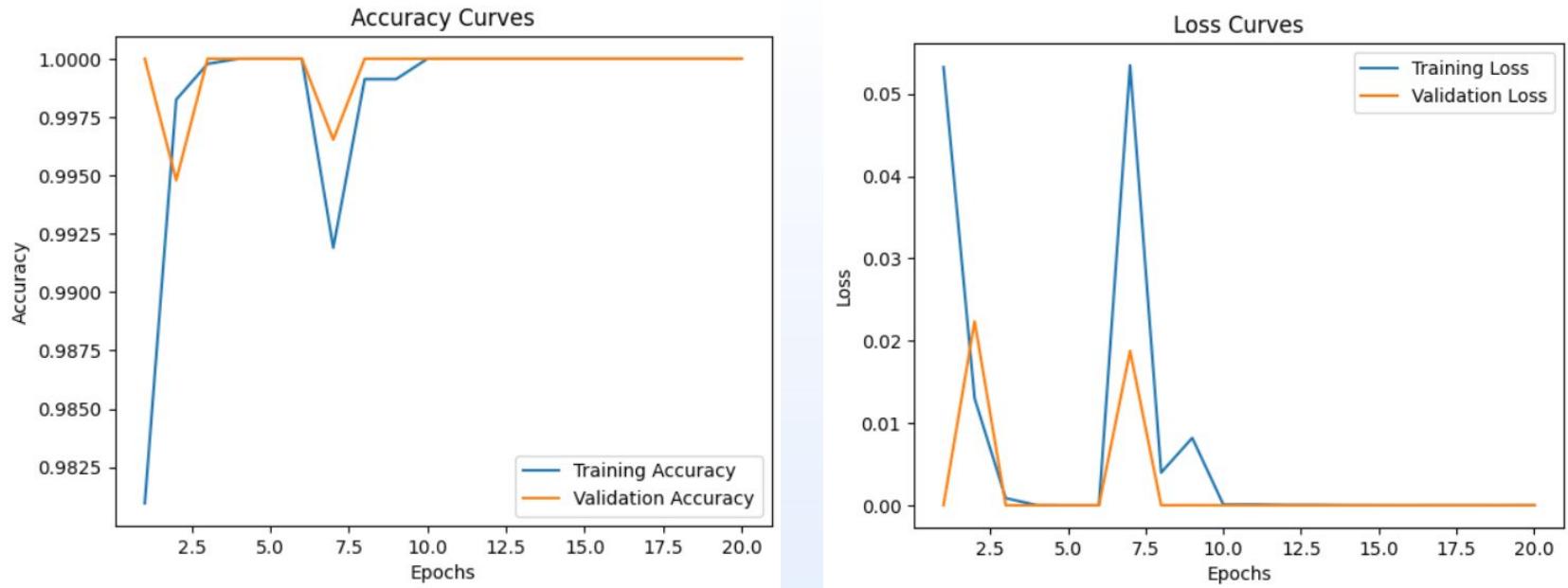
- Frameworks and Libraries
 - Tensorflow
 - keras
- Model Architecture Summary
 - Conv2D
 - MaxPooling2D
 - Flatten
 - Dense
 - Dropout
- Loss Function and Optimization
 - Adam



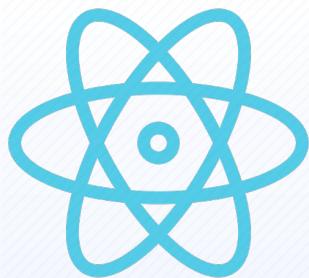
06. Result Comparison

```
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



Frontend & Backend - Technologies



React Native



Flask



Firebase



FORMIK

Frontend & Backend - Code

The screenshot shows a code editor interface with the following details:

- Project Structure:** The left sidebar displays the project structure under "EXPLORER". It includes "OPEN EDITORS" (index.js), "CDAP-PROJECT-FRONTEND" (expo, assets, components, node_modules, screens, AMD-Classification, AMD-Detection/SubScreens), and "services" (gitignore, App.js, app.json, babel.config.js, metro.config.js, Navigation.js, package-lock.json, package.json). Other files like ReadMe and yarn.lock are also listed.
- File Content:** The main area shows the content of index.js. The code handles navigation between screens, manages patient eye disease data, and performs file uploads to Firebase.

```
index.js // cdap-project-frontend

// Import statements
import { StyleSheet, View, Dimensions, ImageBackground } from "react-native";
import { Image, Text } from "react-native";
import React, { useEffect } from "react";
import { SafeAreaView } from "react-native-safe-area-context";
import { TouchableOpacity } from "react-native-gesture-handler";
import { useNavigation } from "react-navigation/native";
import axios from "axios";
import { getFirestore, doc, updateDoc } from "firebase/firestore";
import { db, uploadToFirebase } from "./services/firebase";
import Safe from "../../../../assets/Shield-Done.png";
import Risk from "../../../../assets/Shield-Fail.png";
import TopBar from "../../../../Components/Layouts/TopBar";
import AsyncStorage from "@react-native-async-storage/async-storage";

// Function definitions
const AMD_Detection = ({ route }) => {
  const { image } = route.params;
  const { imageType } = route.params;
  const { docID } = route.params;

  const navigation = useNavigation();

  const [result, setResult] = React.useState("");
  const [status, setStatus] = React.useState("");
  const [imageURL, setImageURL] = React.useState("");
  const [doctorId, setDoctorId] = React.useState("");

  const docRef = doc(db, "Patients", docID);

  const uploadImage = async () => {
    let patientEyeDisease;

    if (imageType === "OCTM") {
      patientEyeDisease = "Age-Related Macular Degeneration";
    } else if (imageType === "FUNDUS") {
      patientEyeDisease = "Diabetic Retinopathy";
    }

    const fileName = image.split(".").pop();
    const uploadResponse = await uploadToFirebase(image, fileName, (v) => {
      console.log("v: ", image, fileName, v);
    });
    console.log("uploadResponse: ", uploadResponse);
    setImageURL(uploadResponse.downloadUrl);

    const patientDataRef = {
      patientEyeDisease: patientEyeDisease,
      img_url: uploadResponse.downloadUrl,
      doctorId: doctorId,
      status: "Negative",
    };
  }
}

// Main export
export default AMD_Detection;
```

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure with files like `main.py`, `models`, `Dockerfile`, and `requirements.txt`.
- Editor View:** Displays the content of `main.py`. The code implements a Flask API endpoint for AMD detection and classification.
- Status Bar:** Shows the file name as `main.py - TestPython`, the status as `You, 29 seconds ago`, the line number as `Ln 1, Col 1`, and the space usage as `Spaces: 4`.

```
main.py 3, M x
main.py > ...
@app.route('/amd/detection', methods=['POST'])
def amd_detection():
    try:
        data = request.get_json()
        image_url = data['image_url']

        # Download the image from the URL
        response = requests.get(image_url)

        if not response.content:
            raise ValueError("Empty image content")

        # Load the image using TensorFlow's load_img
        img = tf.keras.preprocessing.image.load_img(BytesIO(response.content), color_mode='grayscale', target_size=(224, 224))

        # Convert the image to a numpy array
        x = tf.keras.preprocessing.image.img_to_array(img)

        # Reshape the array to match the input shape of the model
        x = tf.expand_dims(x, axis=0)

        # Normalize the pixel values to be between 0 and 1
        x /= 255.0

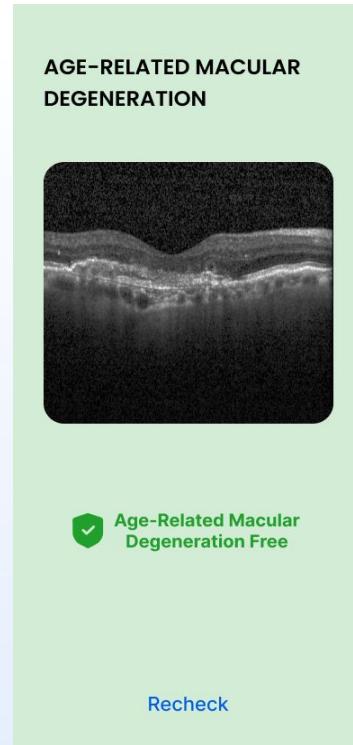
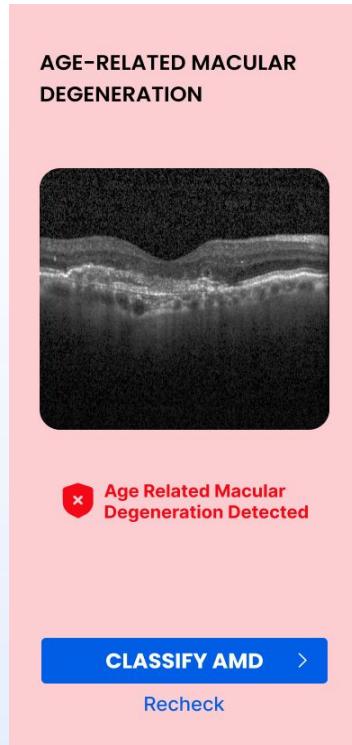
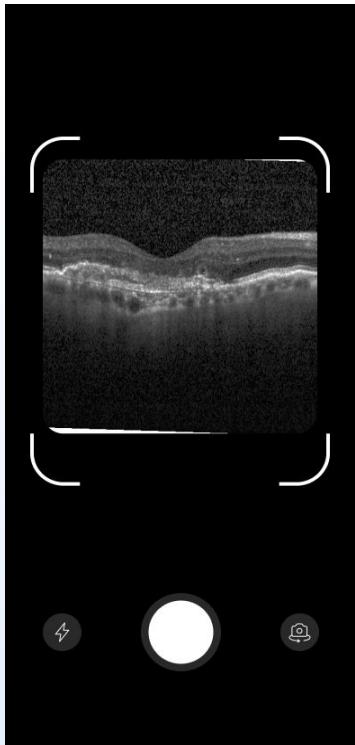
        # Use the model to make a prediction on the image
        prediction = model_amd.predict(x)

        # Print the predicted class label
        if prediction > threshold_amd:
            result = 'AMD Detected'
        else:
            result = 'NORMAL'

        # Return the prediction as a JSON response
        return jsonify({"result": result})
    except Exception as e:
        # Log the error for debugging
        logging.error(str(e))
        return jsonify({"error": "Failed to process the image."}), 400

@app.route('/amd/classification', methods=['POST'])
def amd_classification():
    try:
        data = request.get_json()
        image_url = data.get('image_url')
```

07. User Interfaces



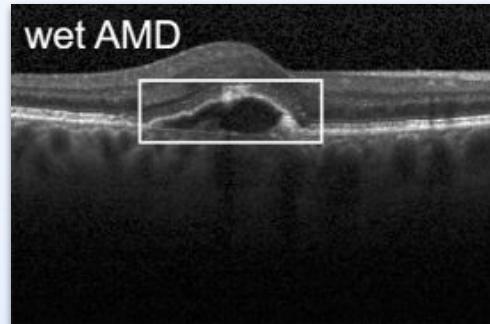
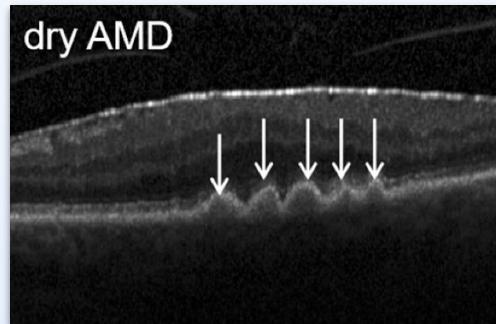


IT20172978 | Kariyawasam K. G. P. C.

Classification of Age-related Macular Degeneration using Retinal OCT Images

Introduction

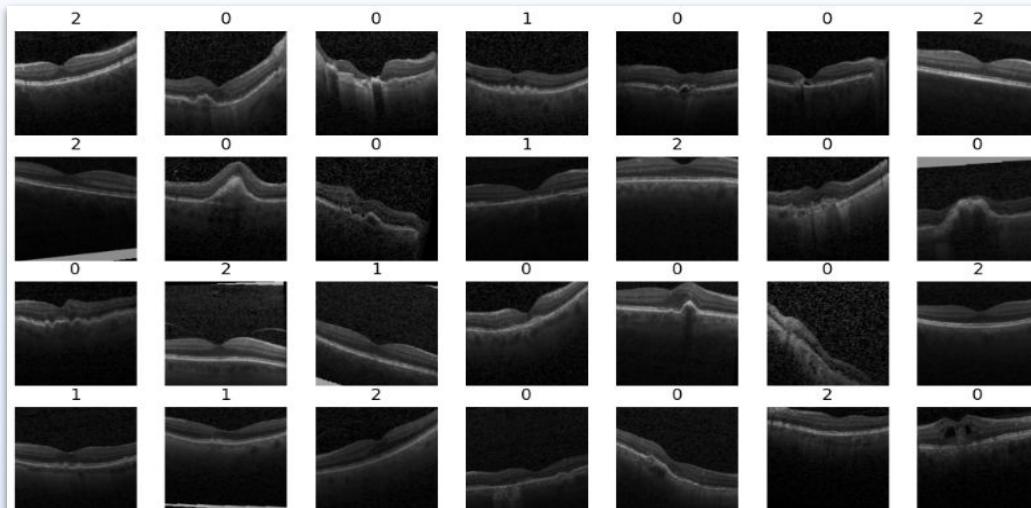
- Develop a novel DL model to accurately detect and classify wet and dry AMD based on a single OCT image while providing annotations for the affected regions.



Datasets

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

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

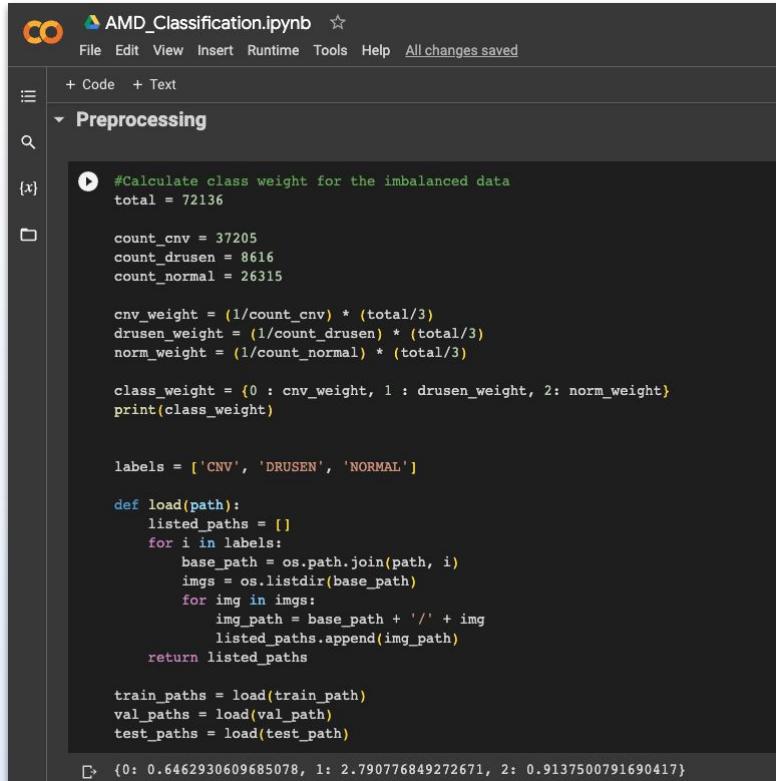


Model Implementation - Technologies

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



Model Implementation - Code



AMD_Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Preprocessing

```
#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)

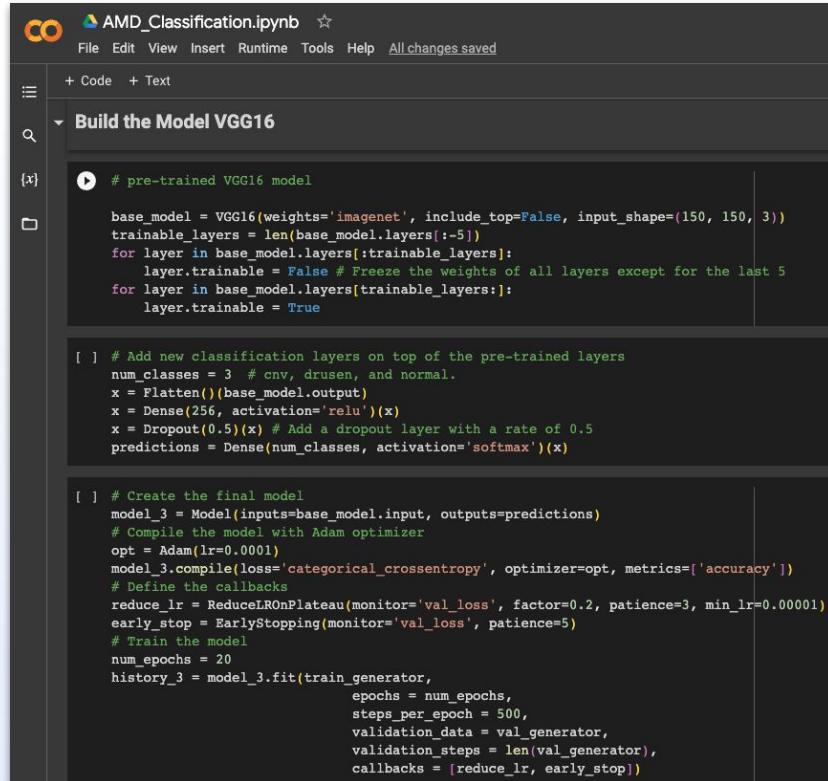
class_weight = {0 : cnv_weight, 1 : drusen_weight, 2: norm_weight}
print(class_weight)

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

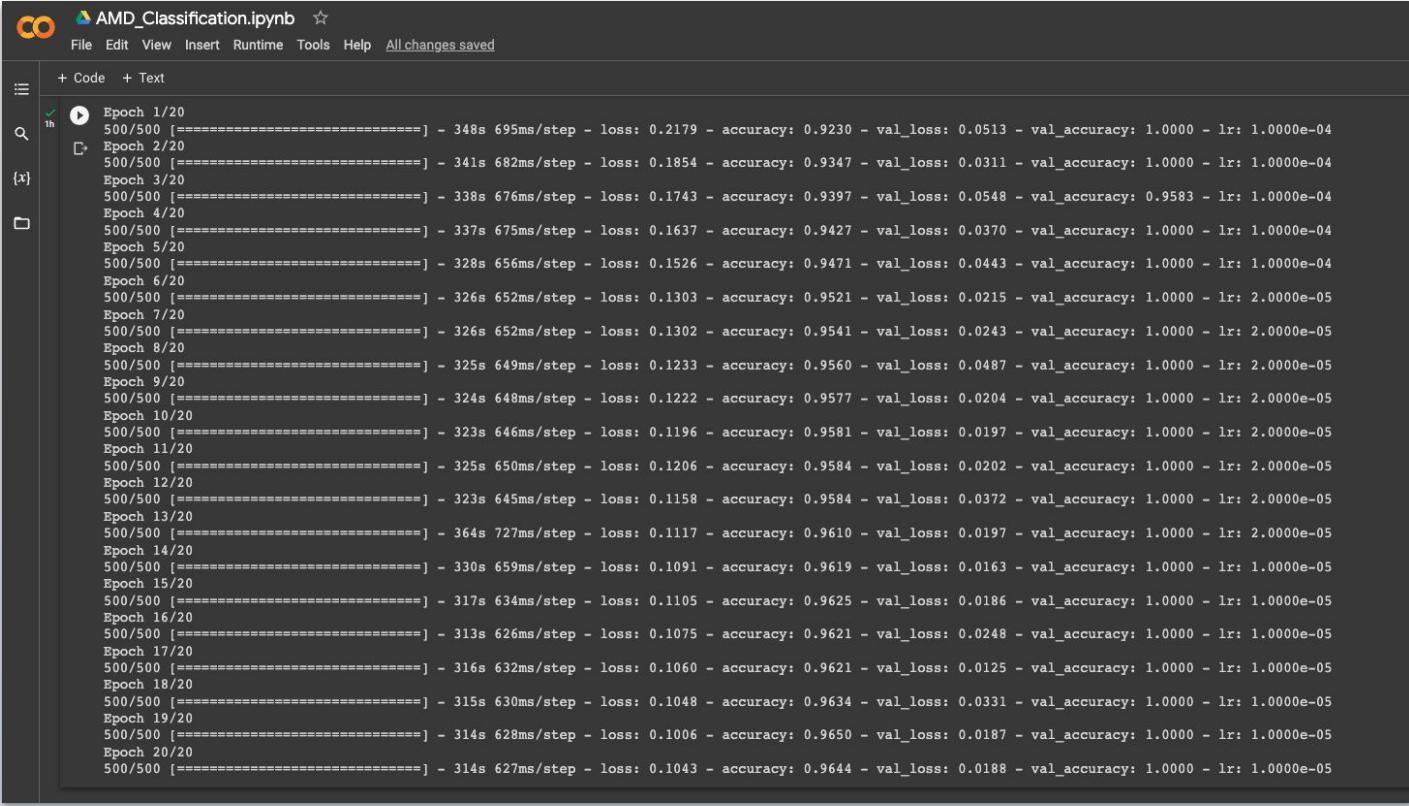
```
# pre-trained VGG16 model

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[:-5]:
    layer.trainable = False # Freeze the weights of all layers except for the last 5
for layer in base_model.layers[-5:]:
    layer.trainable = True

# 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)

# 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 = 50,
    validation_data = val_generator,
    validation_steps = len(val_generator),
    callbacks = [reduce_lr, early_stop])
```

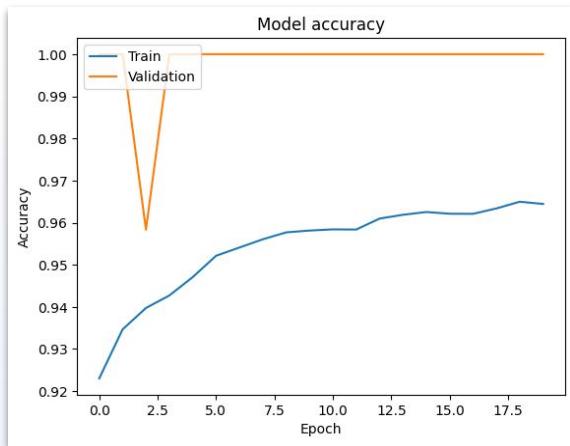
Result Comparison



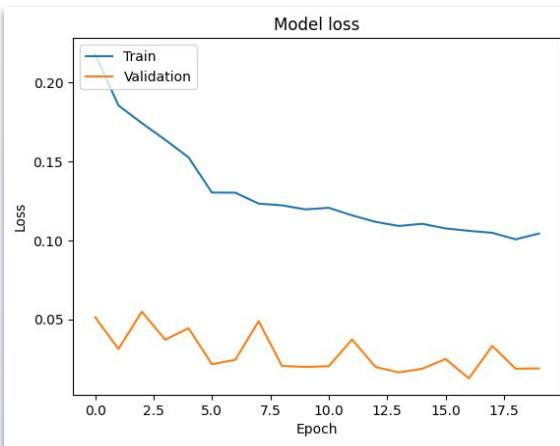
The screenshot shows a Jupyter Notebook interface with the title "AMD_Classification.ipynb". The notebook contains a single code cell displaying a log of training epochs. The log shows 500 epochs of training, with each epoch taking approximately 300ms per step. The accuracy and validation accuracy remain very high, near 1.0, throughout the process.

```
+ 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
```

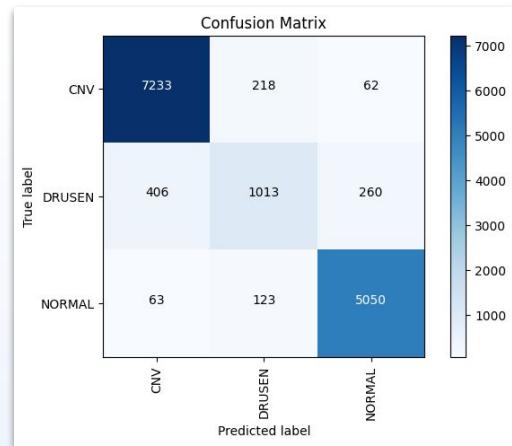
Evidence



Plot training & validation accuracy values



Plot training & validation loss values



Confusion Matrix

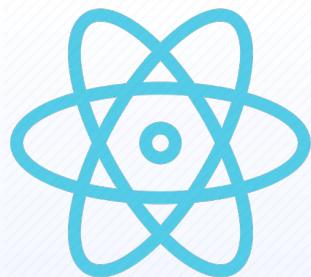
```
Train accuracy: 0.9542208164930344
Train loss: 0.13051368705928326

Validation accuracy: 1.0
Validation loss: 0.024401552975177765

Test accuracy: 0.9820936918258667
Test loss: 0.05077371001243591
```

Train, Validation, and Test Accuracies

Frontend & Backend - Technologies



React Native



Flask



Firebase



FORMIK

Frontend & Backend - Code

AmdClassificationSubPage.js – cdp-project-frontend

EXPLORER

- OPEN EDITORS
- JS AmdClassificationSubPage.js scre... M
- CDAP-PROJECT-FRONTEND
- .expo
- assets
- components
- node_modules
- screens
- AMD-Classification/SubScreens
- JS AmdClassificationSubPage.js M
- JS index.js
- > AMD-Detection
- > DR-Detection
- > DR-Severity
- JS FormikRegister.js
- JS GetImageScreen.js
- JS HomeScreen.js
- JS ImagePreview.js
- JS ImageTypeScreen.js
- JS LoginScreen.js
- JS PatientInfoForm.js
- JS RegisterScreen.js
- JS viewDisease.js
- > services
- .gitignore
- JS App.js
- { app.json
- B babel.config.js
- JS metro.config.js
- JS Navigation.js
- { package-lock.json
- { package.json
- read.ME
- blue yarn.lock

OUTLINE

JS AmdClassificationSubPage.js M X

screens > AMD-Classification > SubScreens > JS AmdClassificationSubPage.js ...

You, 3 minutes ago | 5 authors (PraveenSG and others)

1 > import React, { useEffect, useState } from "react"; PraveenSG, 2 months ago

21 // Define a functional component for the AMD Classification SubPage

22 const AmdClassificationSubPage = ({ route }) => {

23 // State variables for controlling modals and storing data

24 const [isModalVisible1, setIsModalVisible1] = useState(false);

25 const [isModalVisible1, setIsModalVisible1] = useState(false);

26 const [isModalVisible2, setIsModalVisible2] = useState(false);

27 const [classificationResult, setClassificationResult] = useState();

28 const [note, setNote] = useState("");

29 const [patientData, setPatientData] = useState();

30 const [isLoading, setIsLoading] = useState(false);

31 // Extract data from the route parameters

32 const id = route.params.docID;

33 const { image } = route.params;

34 const { imageType } = route.params;

35 // Functions to toggle modals

36 const toggleModal1 = () => {

37 setModalVisible1(!isModalVisible1);

38 };

39 const toggleModal1 = () => {

40 setModalVisible1(!isModalVisible1);

41 };

42 const toggleModal2 = () => {

43 setModalVisible2(!isModalVisible2);

44 };

45 const toggleModal2 = () => {

46 setModalVisible2(!isModalVisible2);

47 };

48 // useEffect to fetch AMD classification result when the component mounts

49

50 const data = {

51 image_url: image,

52 };

53 axios

54 .post(

55 imageType === "OCT" ? "http://127.0.0.1:5000/amd/classification" : "",

56 data

57).then((response) => {

58 setClassificationResult(response.data.result);

59 })

60 .catch((error) => {

61 console.log("This is Error", error);

62 })

63 .catch((error) => {

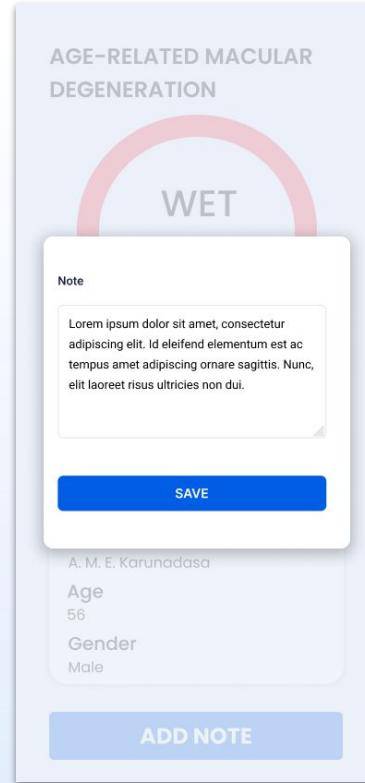
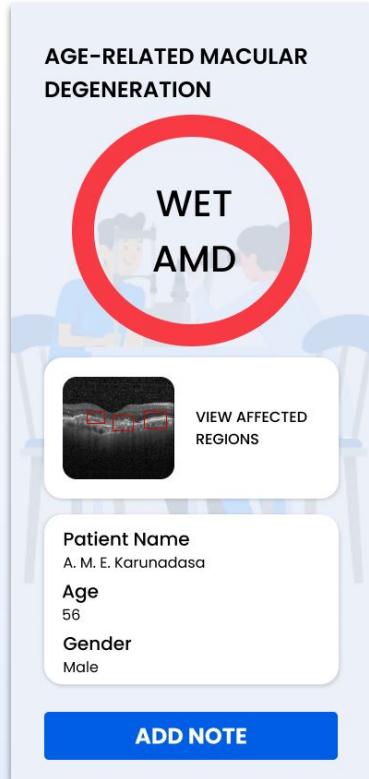
64 console.log("This is Error", error);

65 })

66 }, []);

67 // useEffect to fetch patient data when the component mounts

User Interfaces





Demonstration

Eye Care - Mobile Application for Eye Disease Diagnosis

Features and Benefits

- User-Friendly Interface
- Upload or capture Retinal Image
- Detect and Classify using ML algorithms
- Real-Time Results
- Secure Data Handling
- Cross-platform Application

Non-functional Requirements

- Security & Authentication
- Minimized Response Time
- Improved User-experience
- Database Scalability
- Device Compatibility



Thank You !