SAMSUNG

Samsung Innovation Campus

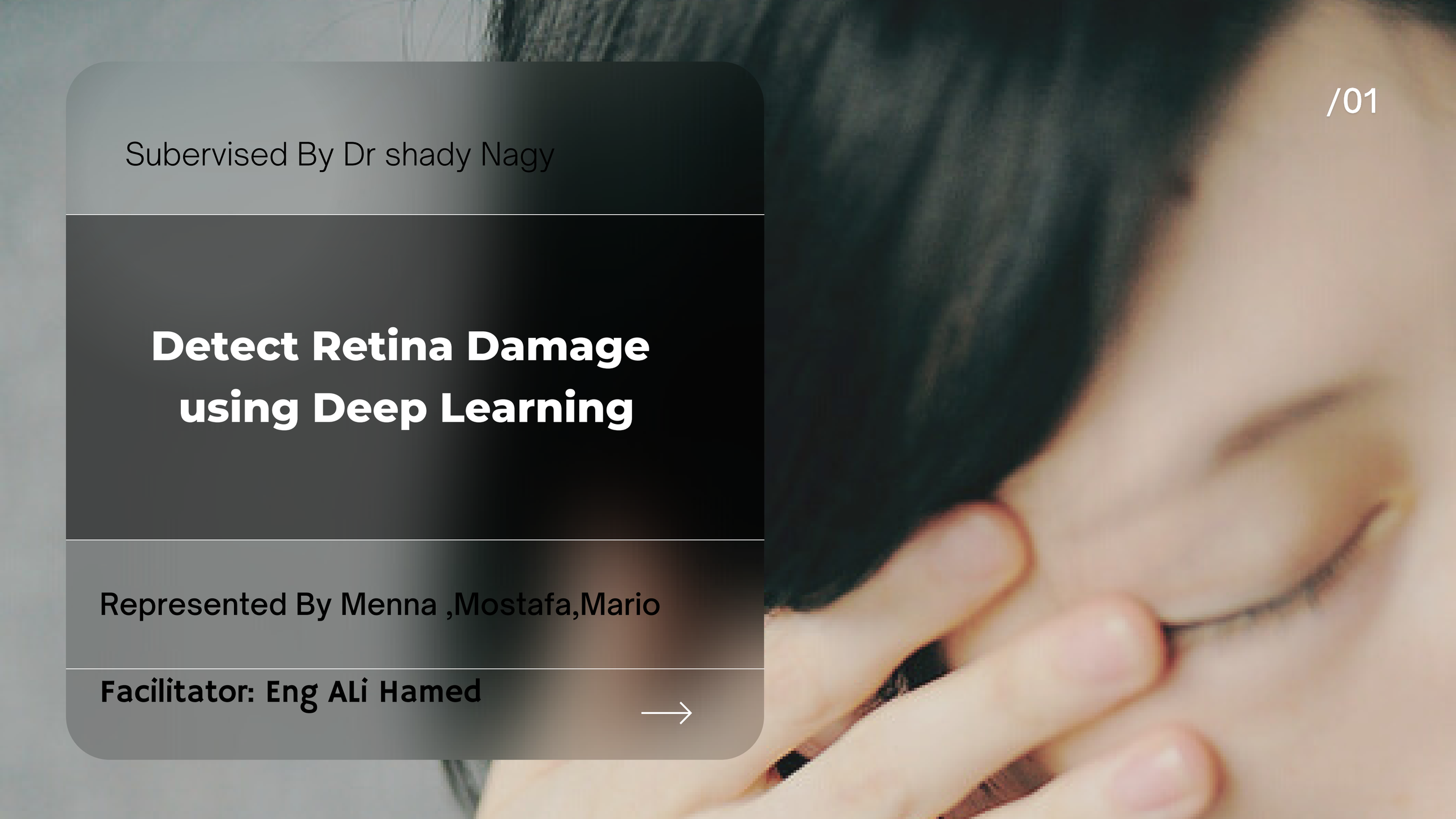
Artificial Intelligence Course

Subervised By Dr shady Nagy

Detect Retina Damage using Deep Learning

Represented By Menna ,Mostafa,Mario

Facilitator: Eng ALi Hamed



Agenda

About

Problem Statement

1

Objective

Visualization

preprocessing

Modeling and Evaluation

Deployment

Conclusion

Recommendation

About :

The retina is the innermost layer of the eye and it features many light-sensitive photoreceptor cells.

These cells detect light and convert it into electrical signals, which travel through the optic nerve to the brain, resulting in sight.

Retinal disorders affect the retina and typically result in visual problems.

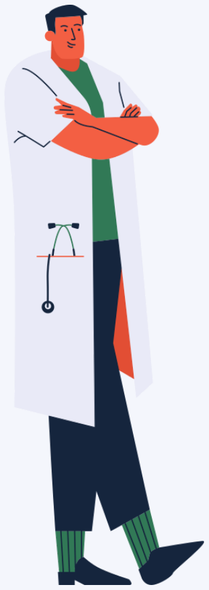
About :

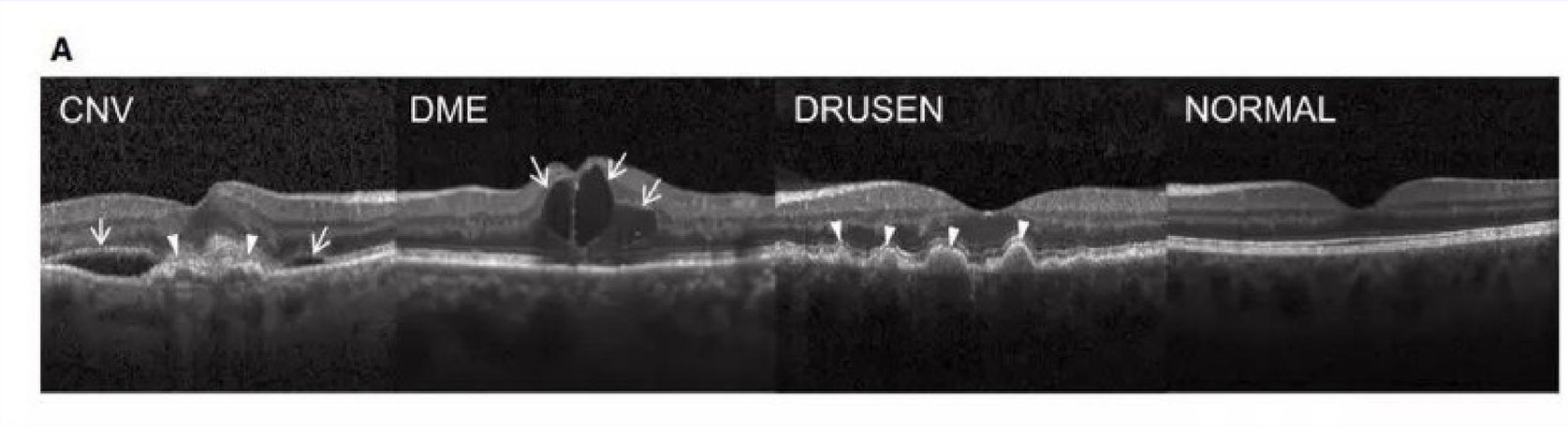
- DRUSEN: Eye contains yellow deposits under the retina
- CNV : Presence of intraretinal or subretinal fluid,
- NORMAL: Eye is in normal condition
- DME: Diabetic macular edema (DME) is a major cause of visual loss in the patients with diabetic retinopathy.

problem statment

- The main objective of this study is to identify what type of retinal disorder the patent have
- Objective:building a system for Retina Damage using deep learning
- Data: contains 84.000 Retinal OCT Images
- Technique: Retinal optical coherence tomography (OCT) is an imaging technique used to capture high-resolution cross sections of the retinas of living patients

Data information



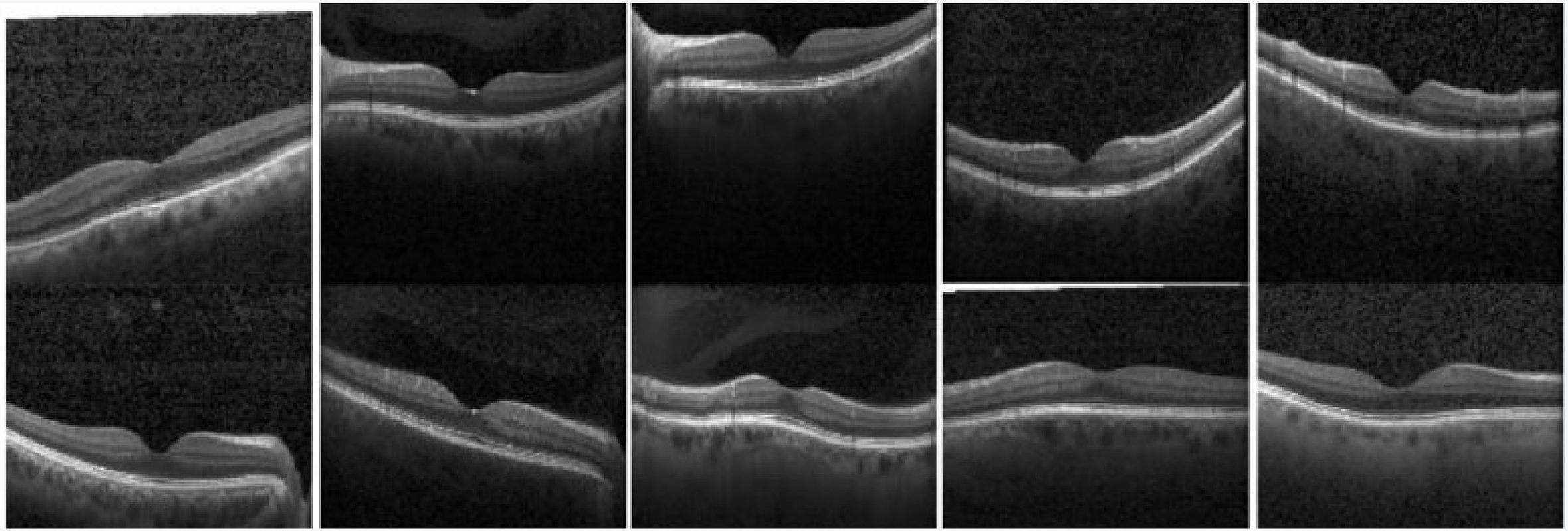


the data have 4 lables (disorders):

- choroidal neovascularization (CNV)
- Diabetic macular edema (DME)
- Multiple drusen(DRusen)
- Normal retina

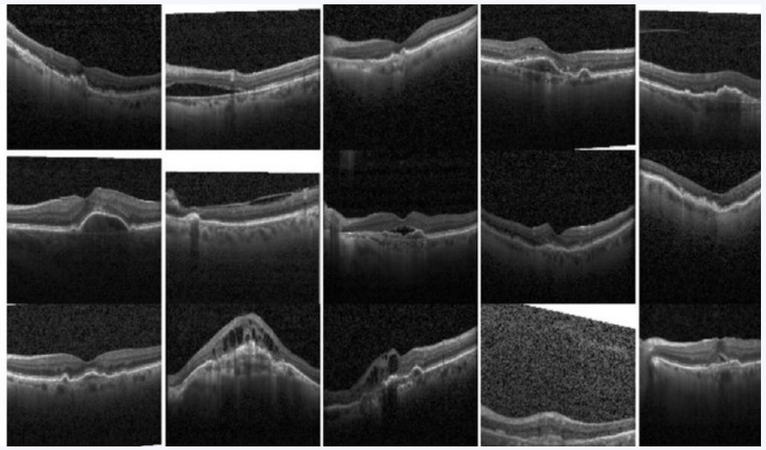
let's explore our data

Normal batch :

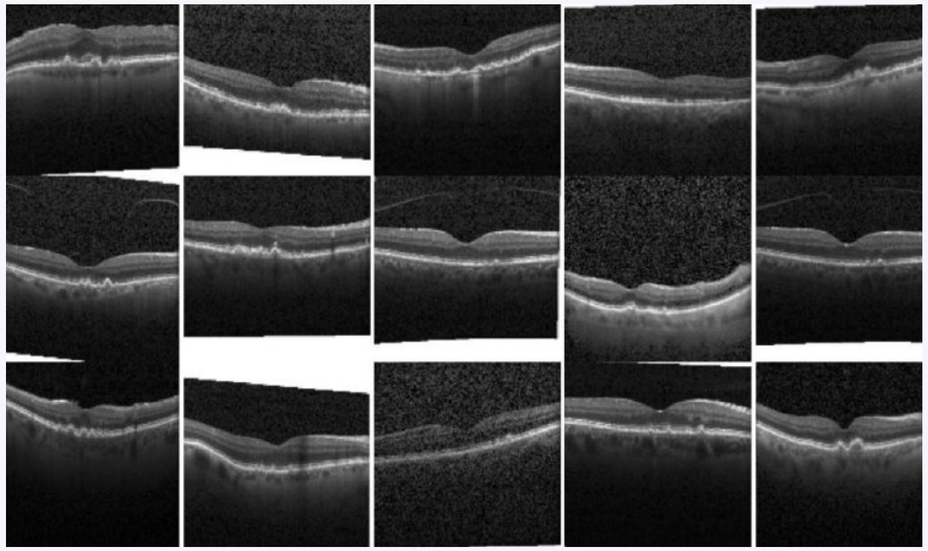


Visualization

Abnormal batch :

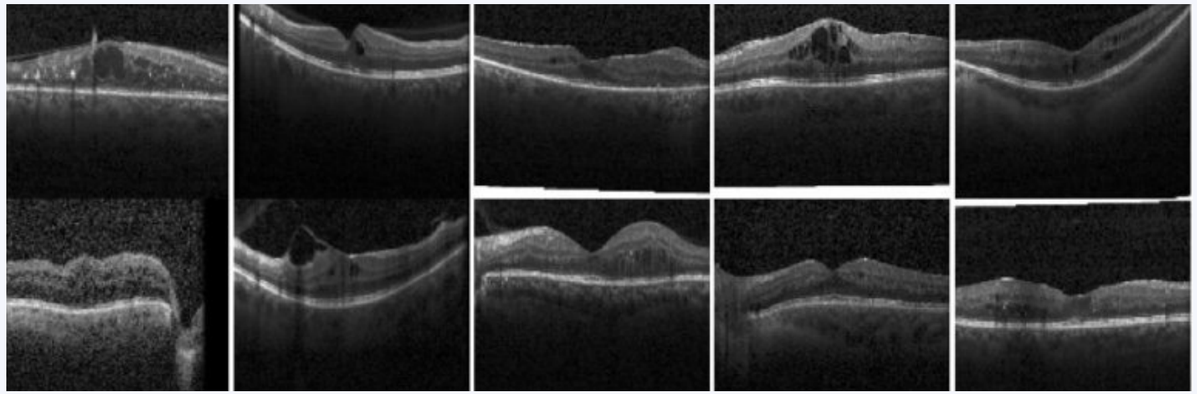


Multiple drusen(DRusen)



CNV

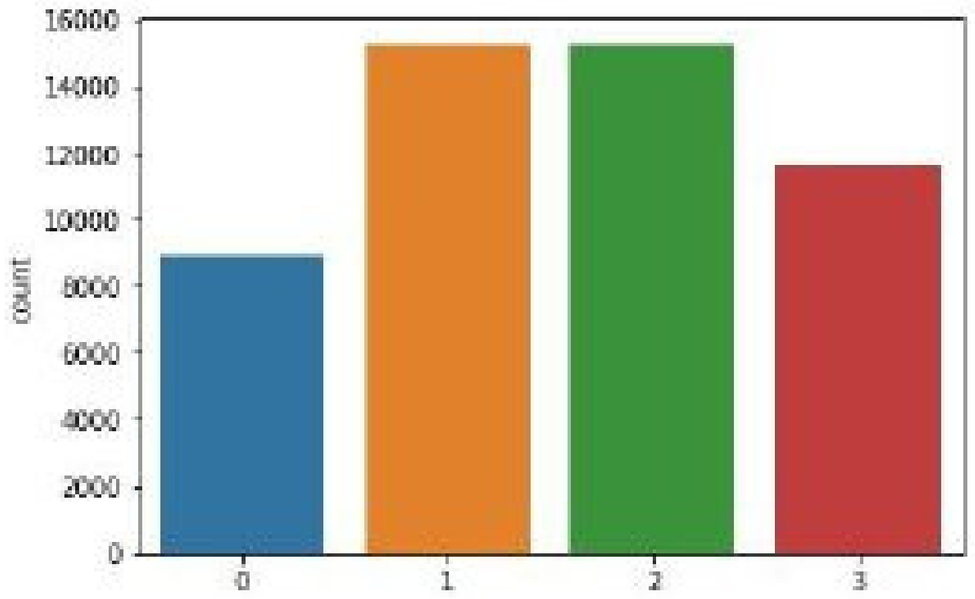
Diabetic macular edema (DME)



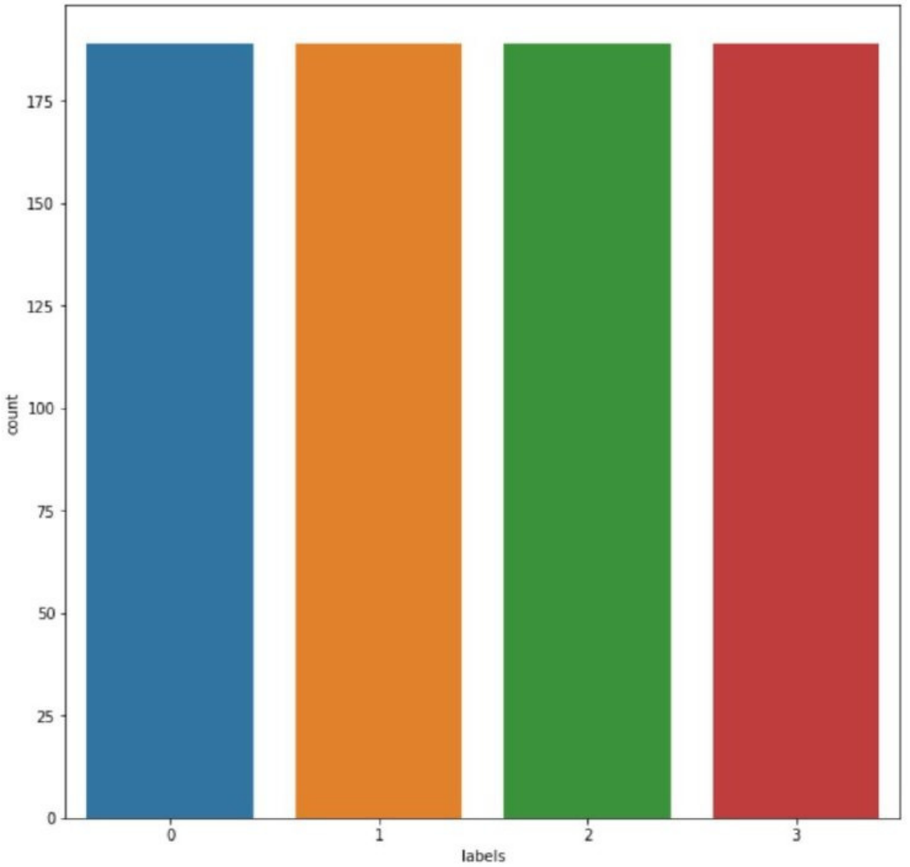
Preprocessing

the classes were imbalanced so we made them have the same level of balance

```
[6]: sns.countplot(labels);
     plt.show()
```



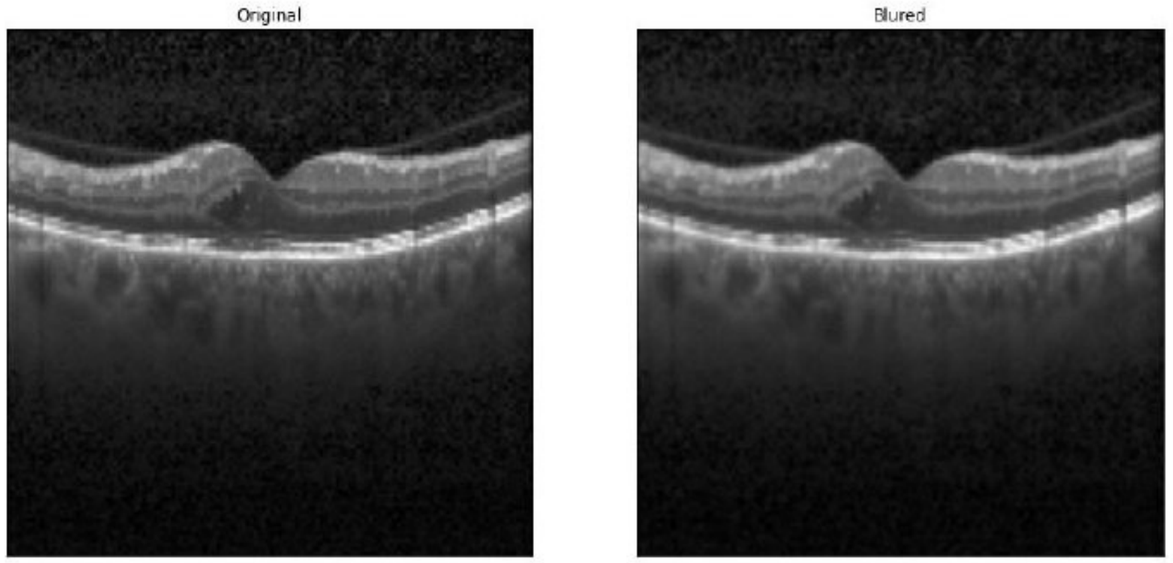

```
{0: 'Normal', 1: 'CNV', 2: 'DME', 3: 'DRUSEN'}
```



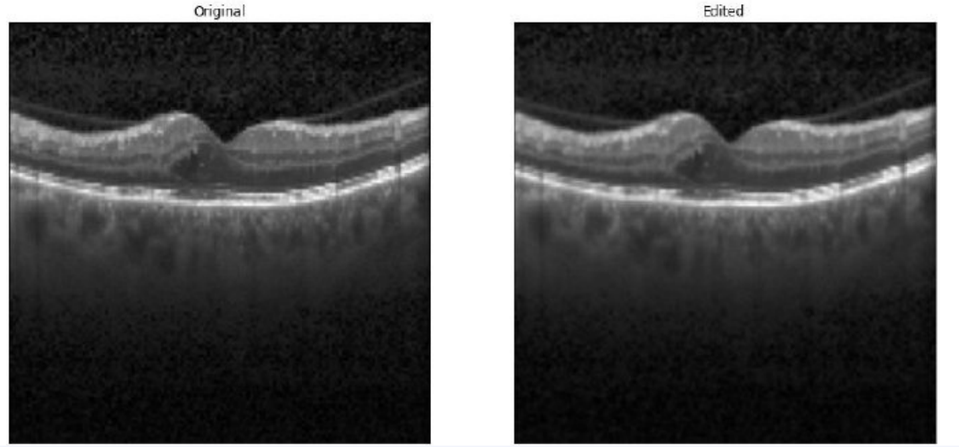
Preprocessing

- we compared between the images before and after applying the filters and as we see we lost a lot of information

```
image = no_noise[1]
display(train_images, res_img1, 'Original', 'Blured')
```

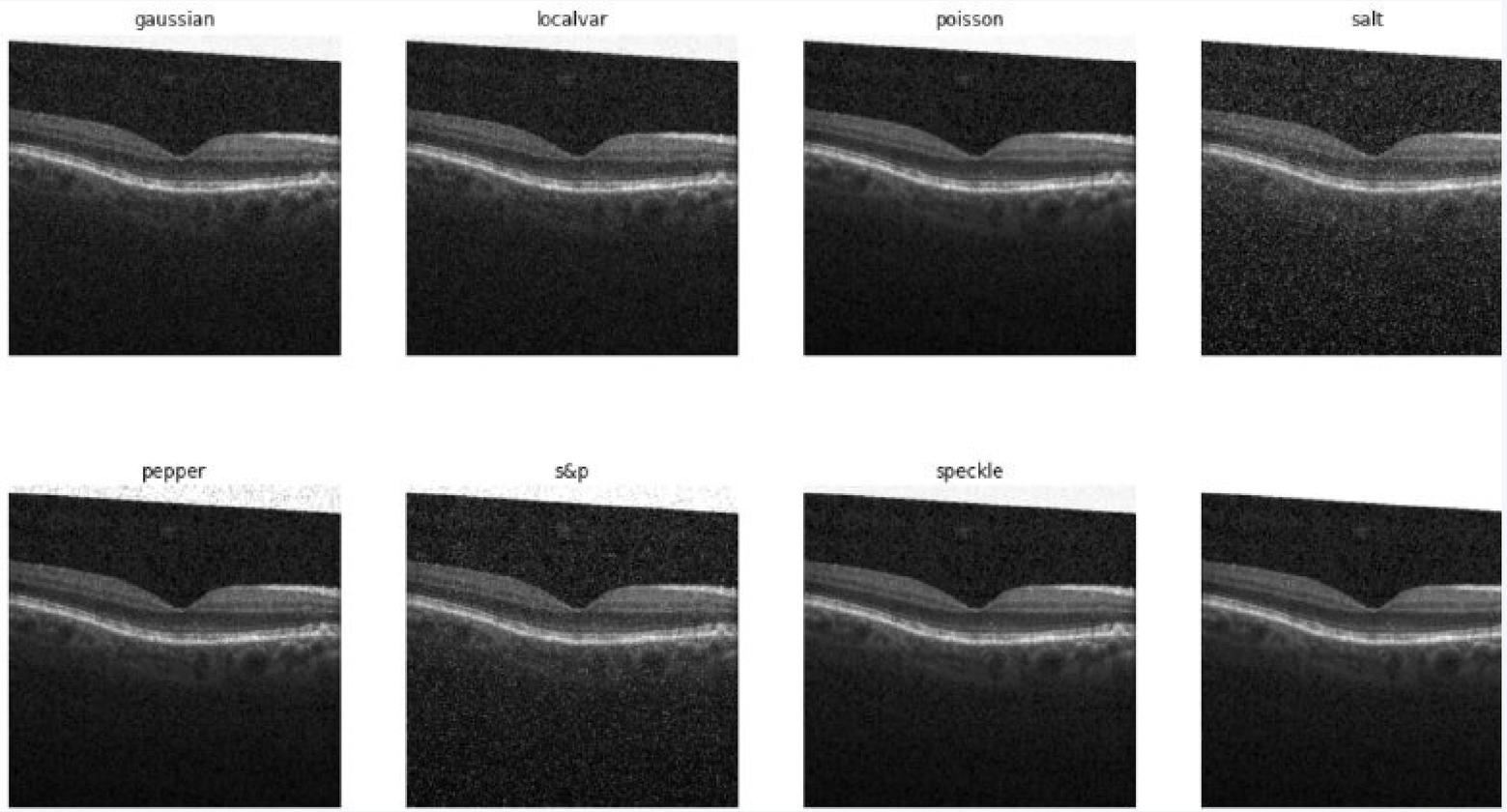



```
def display_one(train_images, title1 = "Original"):
    plt.imshow(train_images[0]), plt.title(title1)
    plt.xticks([]), plt.yticks([])
    plt.show()
# Display two images
def display(train_images, res_img, title1 = "Original", title2 = "Edited"):
    plt.subplot(121), plt.imshow(train_images[0],cmap=plt.cm.gray), plt.title(title1)
    plt.xticks([]), plt.yticks([])
    plt.subplot(122), plt.imshow(res_img[0],cmap=plt.cm.gray), plt.title(title2)
    plt.xticks([]), plt.yticks([])
    plt.show()
display(train_images,res_img1,title1 = "Original", title2 = "Edited")
```



Preprocessing

we compared between the images applying many types of filters :

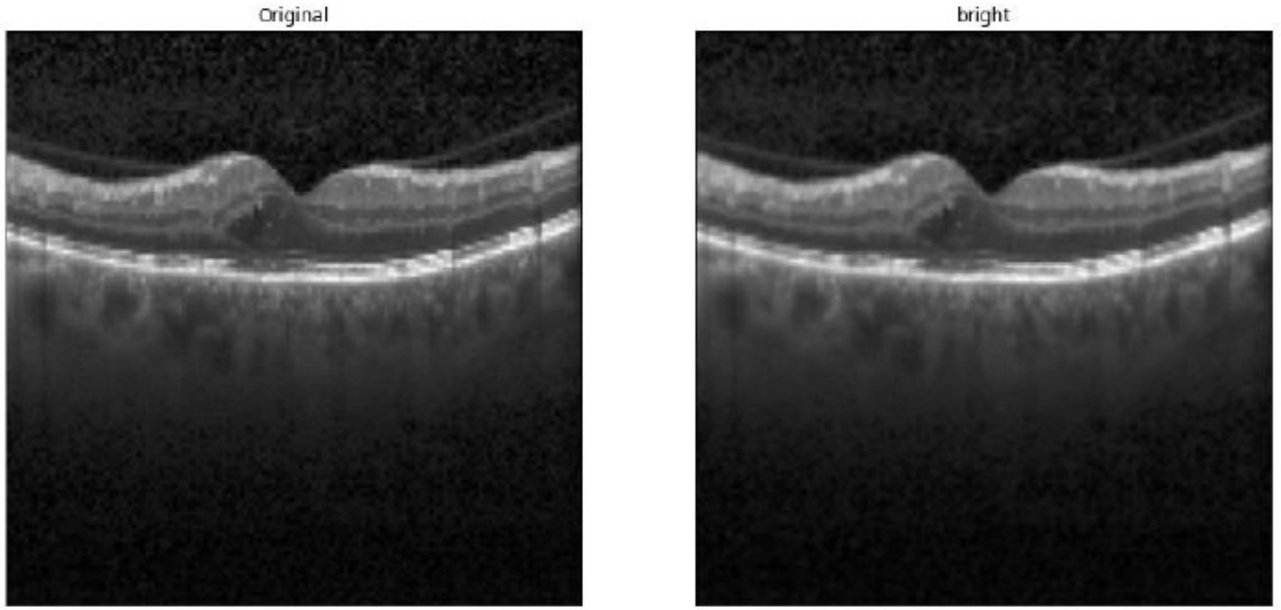


as we see we lost a lot of information

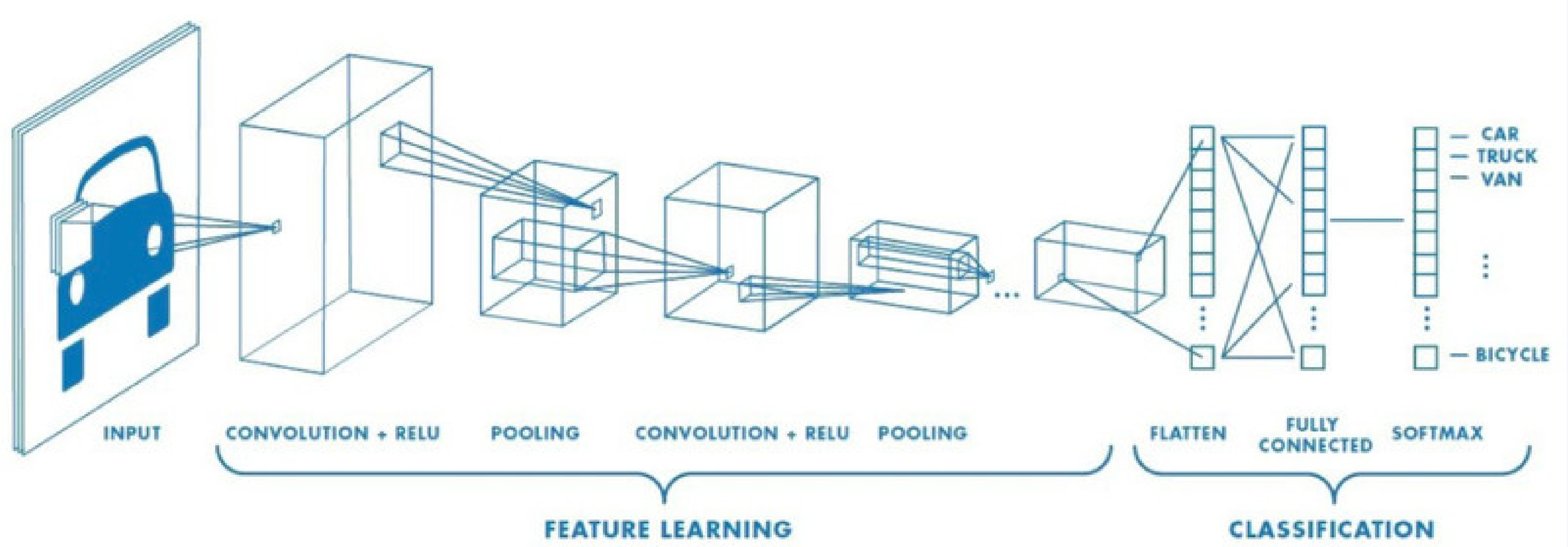
Preprocessing

- we tried to increase the brightness of each photo but we noticed that we missed part of the information so we didnt apply it in our data

In [21]:
```
tf.image.adjust_brightness(
    res_img1, delta=0.1)
display(train_images, res_img1, 'Original', 'bright')
```



Modeling


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Modeling

- We implemented Convolutional Neural Networks (CNN)

- We tried more than one design and they had problems with overfitting and we solved it using dropout layer

- The data was divided into 80% training and 20% testing, then we passd it to the model

Modeling

Model Summary :

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 124, 124, 60)      1560
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 120, 120, 60)      90060
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 117, 117, 30)      28830
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 114, 114, 30)      14430
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 57, 57, 30)        0
_________________________________________________________________
dropout (Dropout)            (None, 57, 57, 30)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 53, 53, 60)        45060
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 50, 50, 30)        28830
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 47, 47, 30)        14430
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 30)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 23, 23, 30)        0
...
Total params: 1,651,424
Trainable params: 1,651,424
Non-trainable params: 0
```

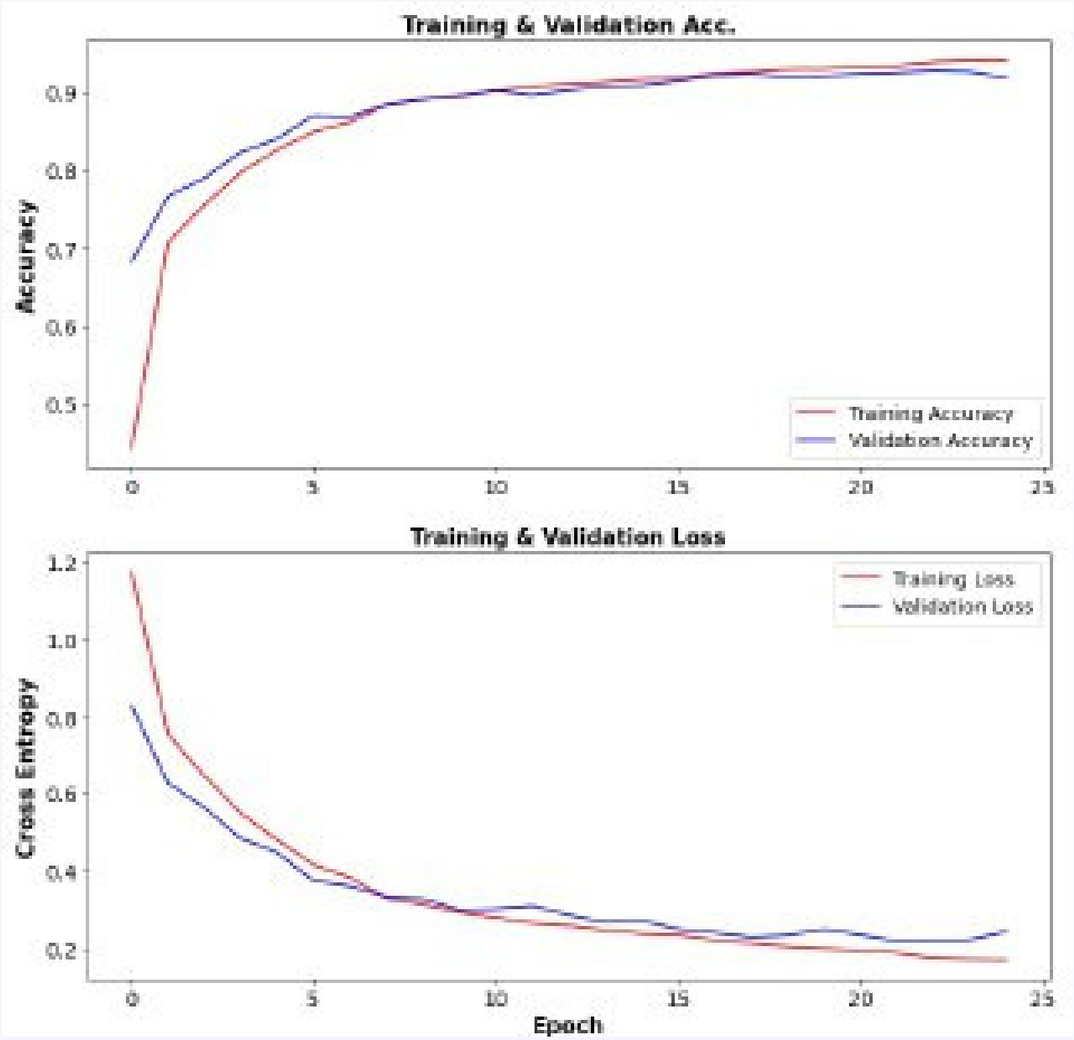
Modeling

Applying model on preprocessed images, give results:

```
Epoch 21/25
112/112 [==============================] - 81s 720ms/step - loss: 0.1727 - accuracy: 0.9392 - val_loss: 0.2174 - val_a
ccuracy: 0.9328
Epoch 22/25
112/112 [==============================] - 81s 720ms/step - loss: 0.1739 - accuracy: 0.9404 - val_loss: 0.2297 - val_a
ccuracy: 0.9230
Epoch 23/25
112/112 [==============================] - 81s 719ms/step - loss: 0.1708 - accuracy: 0.9407 - val_loss: 0.2353 - val_a
ccuracy: 0.9262
Epoch 24/25
112/112 [==============================] - 80s 719ms/step - loss: 0.1589 - accuracy: 0.9446 - val_loss: 0.2215 - val_a
ccuracy: 0.9284
Epoch 25/25
112/112 [==============================] - 80s 718ms/step - loss: 0.1495 - accuracy: 0.9486 - val_loss: 0.2076 - val_a
ccuracy: 0.9334
```

Evaluation

the model was good and it didnt have fluctuations



Evaluation

we tested our model on real photos using this function :

Test cases

```
imge_path1 = "../input/kermany2018/OCT2017 /train/DME/DME-1072015-1.jpeg"  #DME
imge_path2 = "../input/kermany2018/OCT2017 /train/CNV/CNV-1016042-100.jpeg"  #DRUSEN

def predict(img):
    input_img = cv2.imread(img)
    input_img = cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
    input_img_resize = cv2.resize(input_img,(128,128))
    input_img_resize = np.expand_dims(input_img_resize, axis=2)
    input_img_resize = np.expand_dims(input_img_resize, axis=0)
    model_prediction = model.predict(input_img_resize)
    model_prediction = model_prediction.astype(int)
    return model_prediction

print(predict(imge_path1))
print(predict(imge_path2))

# The order of labels
# 'DRUSEN','CNV','NORMAL', 'DME'
```

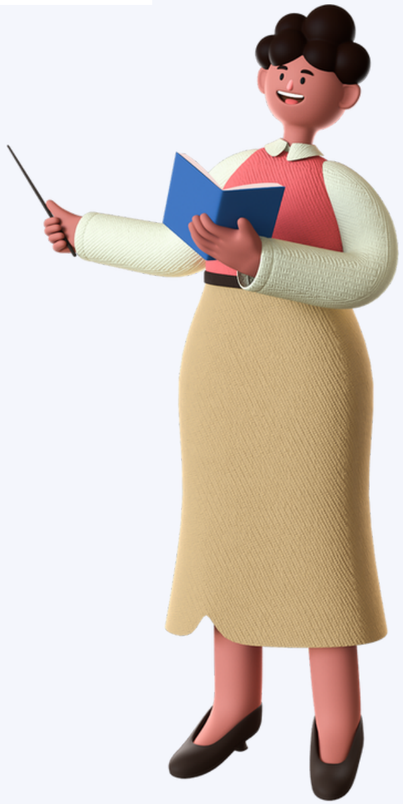
```
[[0 0 0 1]]
[[1 0 0 0]]
```

True prediction for 2 random test cases

Model Outputs and Outcomes

Model Evaluation :

```
248/248 [==============================] - 9s 31ms/step - loss: 0.2181 - accuracy: 0.9317
Test Loss: 0.21813897788524628
Test accuracy: 0.9317263960838318
```



After several attempts, we made a good model design :

- **Our model have a good acc = 93%**
- **we dont have overfitting**
- **We're ready to save our model and create APIs**

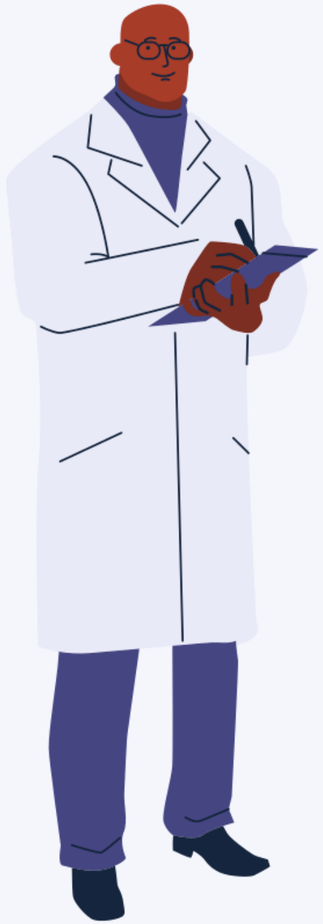
Save Model

How was the model saved :

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights("model.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

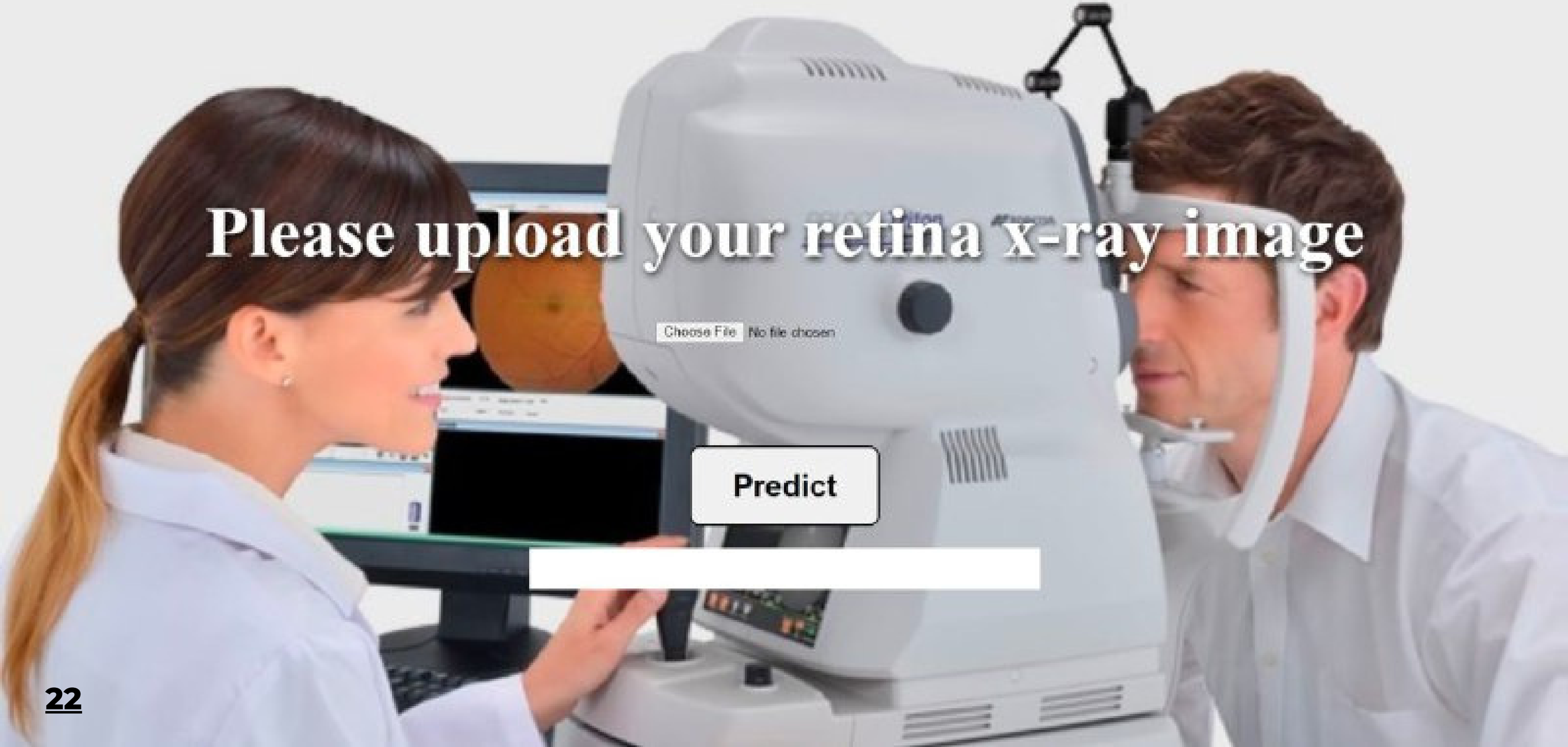
Deployment



Tools used in Deployment :

- **Flask Framework to build Api**
- **Html, Css to build Gui**

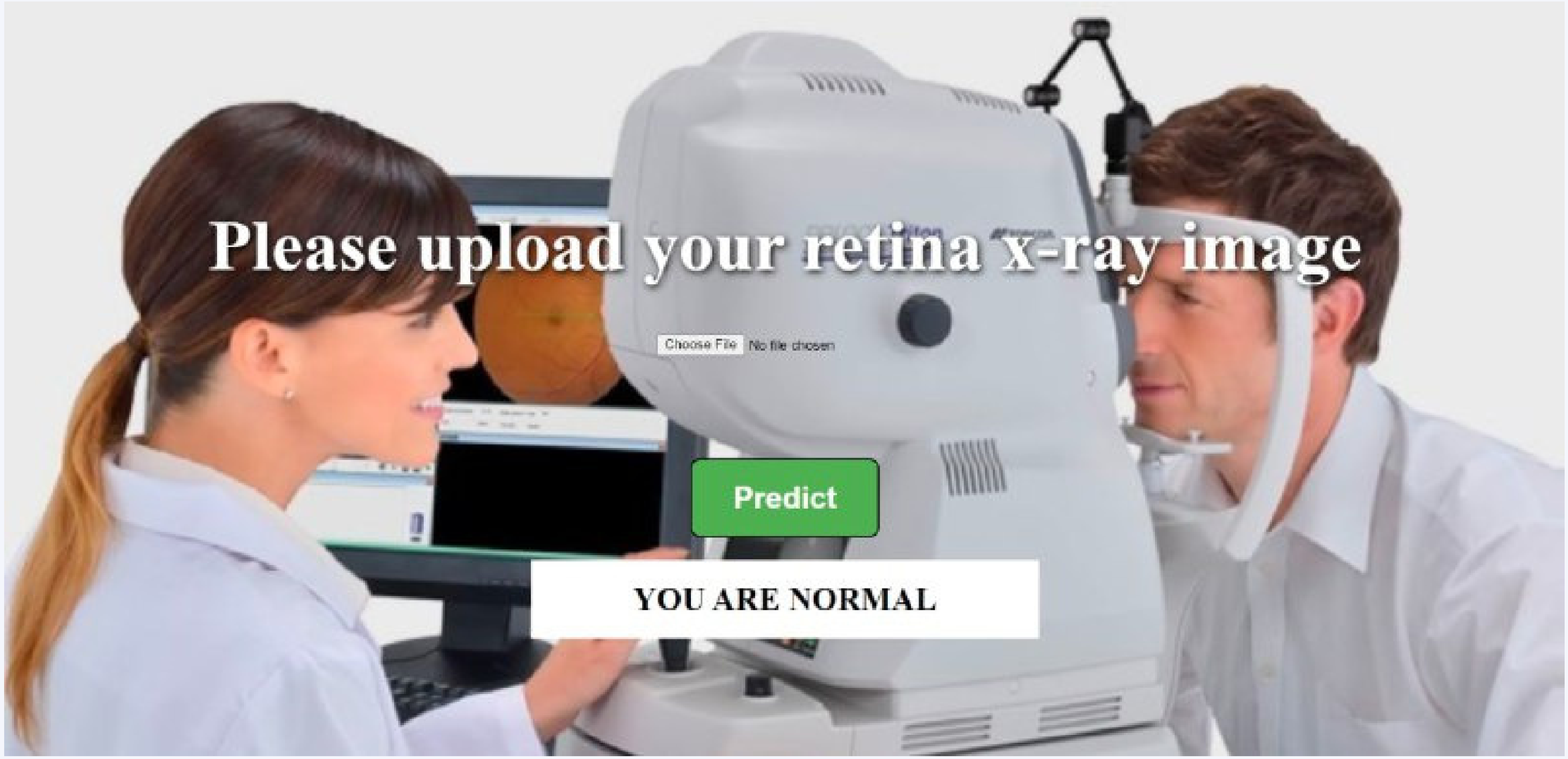
Deployment


Please upload your retina x-ray image
Choose File
No file chosen
Predict

Deployment

- **we upload oct image to predict the type of damage :**

Deployment

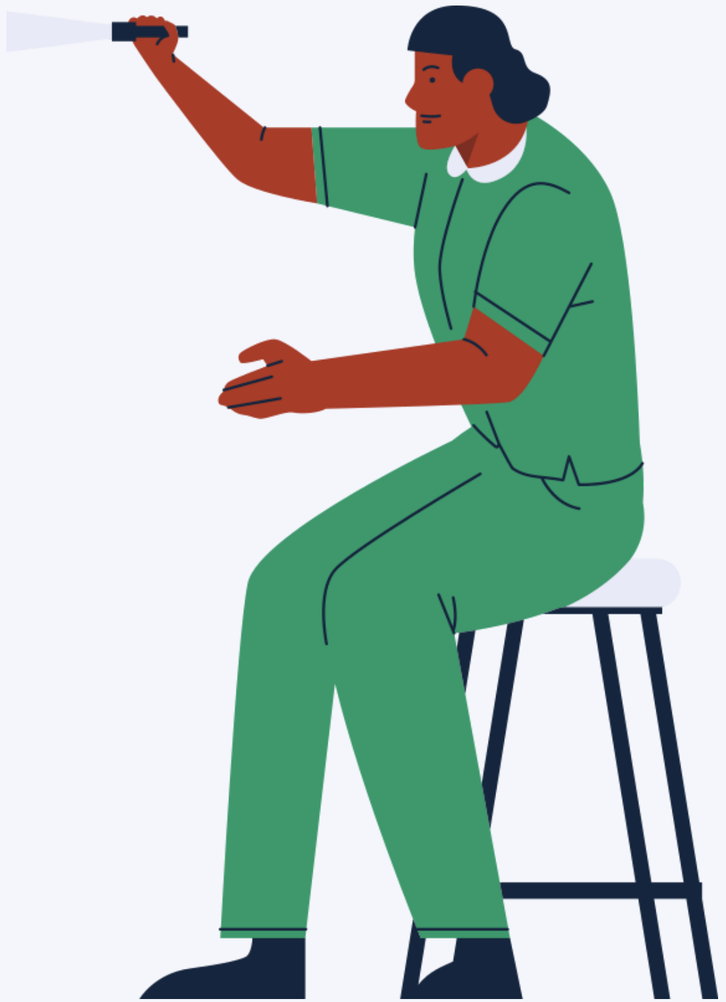


- **the result of the oct image was normal so the patient is good**

Future work

- we tried to use SRGANS but the model needed more memory so in the future we will solve this problem

- we will continue our work with the application but with the description of each damage and what to do to heal from the damage and which behivours he must protect himself from during the damage



GET CONTACT

EMAIL ADDRESS mostafamenna304@gmail.com

PHONE NUMBER 01063557277

EMAIL ADDRESS mostafa.ahraf154@gmail.com

PHONE NUMBER 01283033359

LINKEDIN HTTPS://WWW.LINKEDIN.COM/IN/MENNA-MOSTAFA-1AB0311B7

LINKEDIN https://www.linkedin.com/in/mostafa-ashraf-17913a211/

THANK YOU!

SAMSUNG

Together for Tomorrow!

Enabling People

Education for Future Generations