Name -SHREYAS KUMAR TAH
Internship Program - ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING
Certificate Code-TCRIL01R34
Date of submission - 17-08-2022

Technical Coding Research Innovation, Navi Mumbai,
Maharashtra, India-410206

HEALTHY AND UNHEALTHY EYE DETECTOR

A Case-Study Submitted for the requirement of Technical Coding Research Innovation

For the Internship Project work done during
ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
INTERNSHIP PROGRAM

by
SHREYAS KUMAR TAH (TCRIL01R34)

Rutuja Doiphode
CO-FOUNDER &CEO
TCR innovation.

I. Abstract : Medical health systems have been concentrating on artificial intelligence techniques for speedy diagnosis. However, the recording of health data in a standard form still requires attention so that machine learning can be more accurate and reliable by considering multiple features. The aim of this study is to develop a general framework for detecting and differentiating between a healthy and unhealthy eye. Since artificial intelligence (AI) systems are capable of advanced problem solving, use of such techniques could lead to more objective diagnosis. Although the term 'AI' is commonly used, recent success in its applications to medicine is mainly due to advancements in the sub-field of machine learning, which has been used to automatically classify images and predict medical outcomes. Powerful machine learning techniques have been harnessed to understand nuances in patient data and medical images, aiming for consistent diagnosis and stratification of disease severity. Here the study deals with creating a model using advance machine learning techniques like Neural Networks, etc. to classify eyes between healthy and unhealthy eyes.

II. Introduction : In the near future, the number of patients suffering from eye diseases is expected to increase dramatically due to aging of the population and increasing of eye screen time while digital usage popularly observed among teenagers. In such a scenario, early recognition and correct management of eye diseases are the main objectives to preserve vision and enhance quality of life. Integration of artificial intelligence (AI) and machine learning (ML) in ophthalmology may be helpful at this aim, having the potential to speed up the diagnostic process and simultaneously reduce the human resources required.
AI is a subset of computer science that deals with using computers to develop algorithms that try to simulate human intelligence.

Former MIT professor of AI and computer science Patrick Winston defined AI as "algorithms enabled by constraints, exposed by representations that support models targeted at loops that tie thinking, perception and action together." AI can be divided into four categories, based on the type and complexity of the tasks a system is able to perform. For example, automated spam filtering falls into the most basic class of AI, while the far-off potential for machines that can perceive people's thoughts and emotions is part of an entirely different AI subset. The four types can be classified as follows :

Reactive Machines: able to perceive and react to the world in front of it as it performs limited tasks
Limited Memory: able to store past data and predictions to inform predictions of what may come next
Theory of Mind: able to make decisions based on its perceptions of how others feel and make decisions
Self-Awareness: able to operate with human-level consciousness and understand its own existence

Today artificially intelligent computer systems are used extensively in medical sciences. Common applications include diagnosing patients, end-to-end drug discovery and development, improving communication between physician and patient, transcribing medical documents, such as prescriptions, and remotely treating patients. In this study we would be seeing how we can apply Neural Netwroks , specificly Neural Networks to detect and classifiy the eyes into healthy/ unhealthy. Furthermore, this article will conclude by highlighting the critical importance of dataset and data preprocessing for model building in the creation of ethical, unbiased artificially intelligent systems.

III. Case Study

III.1 Title : Model to predict weather the eye is heallthy/ unheaalthy.

III.2 Objective : To analyze the given dataset and to predict weather the eye is healthy/ unhealthy.

III.3 Tools used : Jupyter Notebook, Numpy, Pandas, Matplotlib, Tensorflow, Convolutional Neural Networks (CNN), Google's pretrained models, Keras.t

III.4 Outcome :

Students are able to:

1. Import the dataset and perform data preprocessing to make it suitable for model building.
2. Get familiar with using Neural Networks specifically with *Convolutional Neural Networks (CNN)*
3. Able to build a model to classify between Healthy and unhealthy eyes.

III.5 Dataset :

Dataset contains 6 types of images including Healthy eyes (42), Cataract eyes (42), Bulging eyes (27), Crossed eyes (114), Glaucomic eyes (31), Uveitis eyes (35).

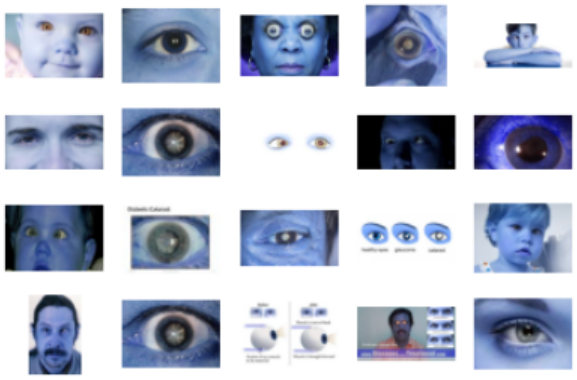
```
In [4]:
vals = [bulging_eyes,cataract_eyes,crossed_eyes,glaucomic_eyes,uveitis_eyes,healthy_eyes]
print("the no of bulging eyes images are:", os.listdir(vals[0]).__len__())
print("the no of cataract eyes images are:", os.listdir(vals[1]).__len__())
print("the no of crossed eyes images are:", os.listdir(vals[2]).__len__())
print("the no of glaucomic eyes images are:", os.listdir(vals[3]).__len__())
print("the no of uveitis eyes images are:", os.listdir(vals[4]).__len__())
print("the no of healthy eyes images are:", os.listdir(vals[5]).__len__())

the no of bulging eyes images are: 27
the no of cataract eyes images are: 42
the no of crossed eyes images are: 114
the no of glaucomic eyes images are: 31
the no of uveitis eyes images are: 35
the no of healthy eyes images are: 42
```

```
In [11]:
df.head(10)
```

Out[11]:

	Filepath	Labels
0	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes
1	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes
2	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes
3	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes
4	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes
5	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes
6	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes
7	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes
8	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes
9	D:\books\ML\DATASETS\Eye Diseases Dataset\Bulg...	Bulging Eyes

```
Out[12]: <function matplotlib.pyplot.show(close=None,
```



III.6 Theory :

III.6.1 Python: Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

III.6.2 Numpy : NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random sitmulation and much more.

III.6.3 Pandas : Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

III.6.4 Matplotlib : Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was

introduced by John Hunter in the year 2002.One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

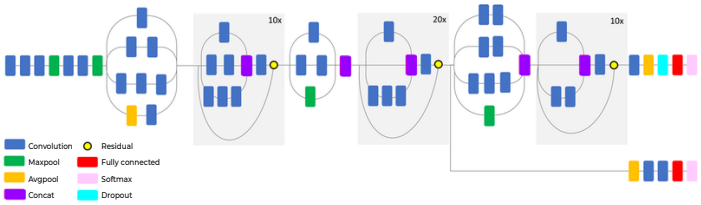
III.6.5 Jupyter Notebooks : Jupyter Notebook is an open-source, web-based interactive environment, which allows you to create and share documents that contain live code, mathematical equations, graphics, maps, plots, visualizations, and narrative text. It integrates with many programming languages like Python, PHP, R, C#, etc.

III.6.6 Convolutional Neural Networks : A convolutional neural network (CNN or ConvNet), is a network architecture for deep learning which learns directly from data, eliminating the need for manual feature extraction. CNNs are particularly useful for finding patterns in images to recognize objects, faces, and scenes. They can also be quite effective for classifying non-image data such as audio, time series, and signal data.

III.6.7 Tensorflow : TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

III.6.8 Keras : Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

III.6.9 INCEPTION RESNET V2 : Inception-Res Net-v2 isa convolutional neural network that is trained on more than a million images from the ImageNet database . The network is 164 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.



IV. Model Building : We use Convolutional Neural Networks (CNN) for which dataset is splitted in three sets namely training set, testing set and validation set. Validation Dataset is the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. It provides the first test against unseen data, allowing data scientists to evaluate how well the model makes predictions based on the new data. The data is splitted in the ratio 9:1.

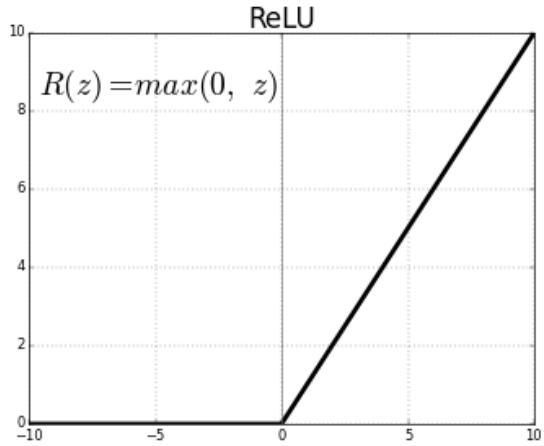
```
In [13]:
1 from sklearn.model_selection import train_test_split
2 Train, Test = train_test_split(df, test_size=0.1, random_state=0)
3 Train_new, valid = train_test_split(Train, test_size=0.1, random_state=0)
4 print(Train.shape, Test.shape, Train_new.shape, valid.shape)

(236, 2) (27, 2) (212, 2) (24, 2)
```

The dataset is in the form of image so it needs to be processed before we fit our model using Image Data Generator class from keras tensorflow. Image data generator class is used to carry out data augmentation where we aim to gain the overall increment in the generalization of the model. Operations such as rotations, translations, shearin, scale changes, and horizontal flips are carried out randomly in data augmentation using an image data generator.

In [14]:
```
train_datagen= ImageDataGenerator(rescale=1./255, rotation_range=45,
zoom_range=0.1, horizontal_flip=True, vertical_flip=True)
test_datagen= ImageDataGenerator(rescale=1./255)

#here we use the image data generator package to do image preprocessi
#We do the same in both test, train and valid too.
```

Further we use flow_from_dataframe from image data generator which takes the data-frame and the path to a directory + generates batches. The generated batches contain augmented/normalized data.
We used activation functions like Relu for our hidden layers of the Neural Network archietect and Softmax for the output layer respectively.

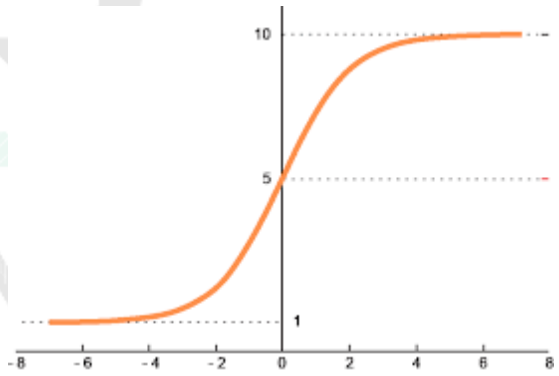
The **rectified linear activation function or ReLU** for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.

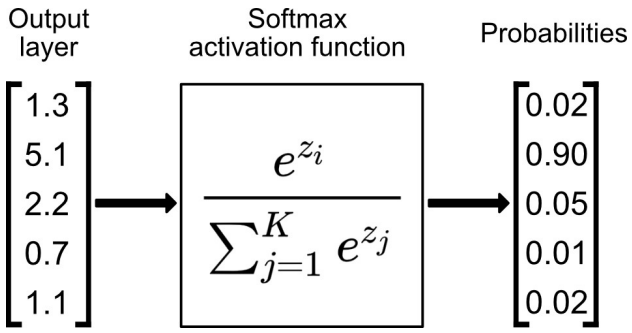


$$\text{ReLU}'(x) = \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases}$$

The usage of ReLU helps to prevent the exponential growth in the computation required to operate the neural network. If the CNN scales in size, the computational cost of adding extra ReLUs increases linearly.

Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. The most common use of the softmax function in applied machine learning is in its use as an activation function in a neural network model. Specifically, the network is configured to output N values, one for each class in the classification task, and the softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the output of the softmax function is interpreted as the probability of membership for each class.





Further we use **Adam Optimizer** in our model because has the best accuracy in enhancing the CNN ability in classification and segmentation.

Adam optimizer involves a combination of two gradient descent methodologies:
Momentum:
This algorithm is used to accelerate the gradient

descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Using averages makes the algorithm converge towards the minima in a faster pace.

$$w_{t+1} = w_t - \alpha m_t$$

where,

$$m_t = \beta m_{t-1} + (1-\beta)\left[\frac{\delta L}{\delta w_t}\right]$$

m_t = aggregate of gradients at time t [current] (initially, m_t = 0)
m_{t-1} = aggregate of gradients at time t-1 [previous]
W_t = weights at time t
W_{t+1} = weights at time t+1
α_t = learning rate at time t
∂L = derivative of Loss Function
∂W_t = derivative of weights at time t
β = Moving average parameter (const, 0.9)

Root Mean Square Propagation (RMSP) : Root mean square prop or RMSprop is an adaptive learning algorithm that tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it takes the 'exponential moving average'.

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t+\varepsilon)^{1/2}} * \left[\frac{\delta L}{\delta w_t}\right]$$

where,

$$v_t = \beta v_{t-1} + (1-\beta) * \left[\frac{\delta L}{\delta w_t}\right]^2$$

W_t = weights at time t
W_{t+1} = weights at time t+1
α_t = learning rate at time t
∂L = derivative of Loss Function
∂W_t = derivative of weights at time t
V_t = sum of square of past gradients. [i.e sum(∂L/∂Wt-1)] (initially, V_t = 0)
β = Moving average parameter (const, 0.9)
ϵ = A small positive constant (10^{-8})

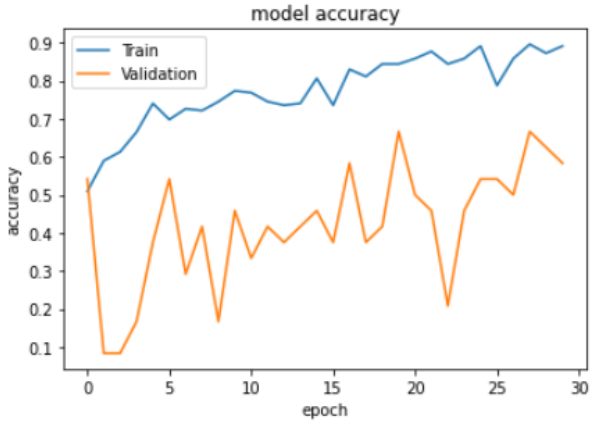
```
In [41]: 1 model.compile(optimizer='Adam',loss='categorical_crossentropy' , metrics=['accuracy'])
         2 history = model.fit(train_gen, validation_data=valid_gen, epochs=30, verbose=2)

Epoch 1/30
14/14 - 126s - loss: 1.4555 - accuracy: 0.5094 - val_loss: 1.6496 - val_accuracy: 0.5417
Epoch 2/30
14/14 - 73s - loss: 1.2258 - accuracy: 0.5896 - val_loss: 34.6479 - val_accuracy: 0.0833
Epoch 3/30
14/14 - 74s - loss: 1.1250 - accuracy: 0.6132 - val_loss: 30.3007 - val_accuracy: 0.0833
Epoch 4/30
14/14 - 75s - loss: 0.9106 - accuracy: 0.6651 - val_loss: 157.9605 - val_accuracy: 0.1667
Epoch 5/30
14/14 - 75s - loss: 0.7422 - accuracy: 0.7406 - val_loss: 1.5029 - val_accuracy: 0.3750
Epoch 6/30
14/14 - 61s - loss: 0.8183 - accuracy: 0.6981 - val_loss: 3.4752 - val_accuracy: 0.5417
Epoch 7/30
14/14 - 66s - loss: 0.7626 - accuracy: 0.7264 - val_loss: 10.4912 - val_accuracy: 0.2917
Epoch 8/30
14/14 - 73s - loss: 0.7489 - accuracy: 0.7217 - val_loss: 3.7805 - val_accuracy: 0.4167
Epoch 9/30
14/14 - 73s - loss: 0.6896 - accuracy: 0.7453 - val_loss: 7.7297 - val_accuracy: 0.1667
Epoch 10/30
14/14 - 74s - loss: 0.6469 - accuracy: 0.7736 - val_loss: 8.3089 - val_accuracy: 0.4583
```

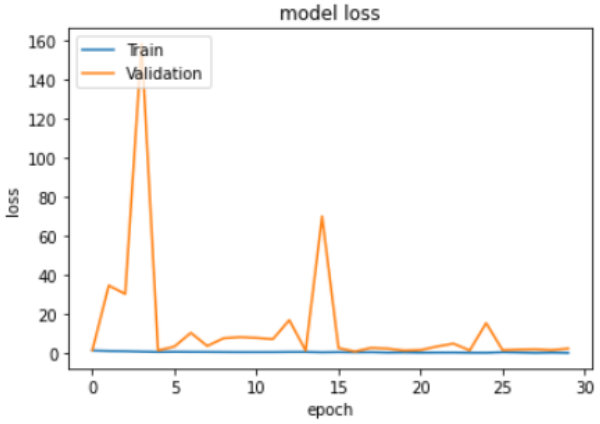
V. Model Summary :

```
Epoch 28/30
14/14 - 59s - loss: 0.2762 - accuracy: 0.8962 - val_loss: 2.1075 - val_accuracy: 0.6667
Epoch 29/30
14/14 - 51s - loss: 0.4074 - accuracy: 0.8726 - val_loss: 1.7091 - val_accuracy: 0.6250
Epoch 30/30
14/14 - 53s - loss: 0.2814 - accuracy: 0.8915 - val_loss: 2.4677 - val_accuracy: 0.5833
```

```
In [42]: 1 model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 3, 3, 1536)        54336736
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1536)              0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               196736
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 774
=================================================================
Total params: 54,534,246
Trainable params: 54,473,702
Non-trainable params: 60,544
```

Accuracy achieved for our model is 89% and the validation accuracy achieved for the model is 58% which means our model is overfitting. Possible reason for overfitting of dataset in our case could be because of the poor collection of training images in the dataset.

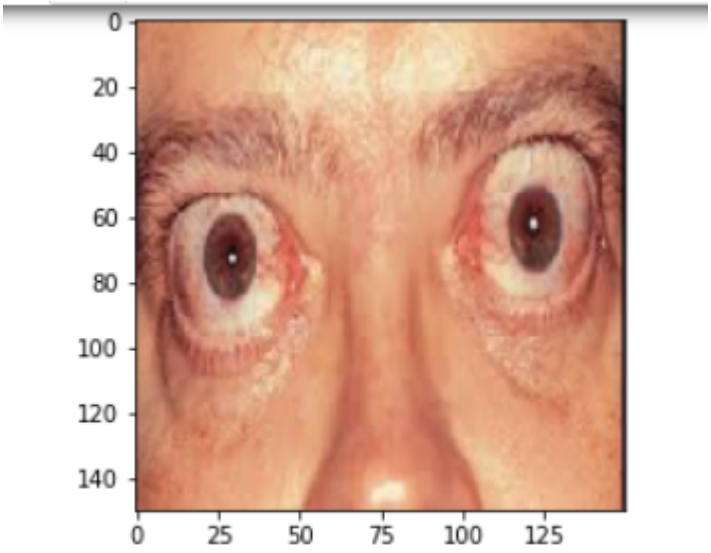




VI. Model Testing :

While testing our model we found that it predicts some of the images accurately whereas it predicts incorrect in other cases as well.

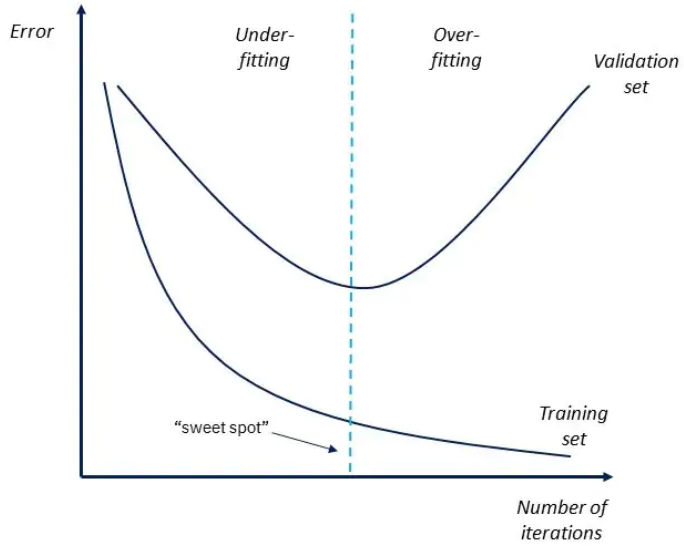
```
In [48]:
path= 'C:/Users/shrey/Documents/Eye Disease Detection Project/Project
for i in os.listdir(path):
    img= image.load_img(path+'//'+ i, target_size=(150,150))
    plt.imshow(img)
    plt.show()

    X= image.img_to_array(img)
    X=np.expand_dims(X, axis=0)
    images= np.vstack([X])

    vals = model.predict(images)
    if vals.all() == 0:
        print("Person has Bulging Eyes")
    elif vals.all() == 1:
        print("Person has Cataract Eyes")
    elif vals.all() == 2:
        print("Person has Crossed Eyes")
    elif vals.all() == 3:
        print("Person has Glaucomic Eyes")
    elif vals.all() == 4:
        print("Person has Healthy Eyes")
    else:
        print("Person has Uveitis Eyes")
```




```
Person has Bulging Eyes
```

VII. Conclusion

Altough the accuracy for our model is 89% still we see that model does not predict accurately for many images. This means that their is a high chance that the model is overfitted. From this we draw possible conclusions such as the reason behind the overfitting of our model might be due to the small amount of dataset in our training set. Overfitting can also occur due to improper yperparameter tuning before training our model. Thus we can improve it by changing/ adding various hyperparameters likes Regularization, Dropouts, etc. to tune our model.



References :

1. Data Driven Approach for Eye Disease Classification with Machine Learning by by Sadaf Malik, Nadia Kanwal, Mamoona Naveed Asghar, Mohammad Ali, A. Sadiq, Irfan Karamat and Martin Fleury. (https://www.mdpi.com/2076-3417/9/14/2789)
2. Artificial intelligence in dry eye disease by Andrea M.Storås, IngaStrümke, Michael A.Riegler, Jako-bGrauslund, Hugo L.Hammer, AnisYazidi, Kjell G.Gundersen, Tor P.Utheim ,Catherine J.Jackson (https://www.sciencedirect.com/science/article/abs/pii/S1542012421001324)
3. Artificial Intelligence in Diabetic Eye Disease Screening by Cheung, Carol Y. PhD*; Tang, Fangyao PhD*; Ting, Daniel Shu Wei MD, PhD†; Tan, Gavin Siew Wei MD†; Wong, Tien Yin MD, PhD (journals.lww.com/apjoo/fulltext/2019/03000)
4. The Impact of Artificial Intelligence and Deep Learning in Eye Diseases: A Review by Raffaele Nuzzi, Giacomo Boscia, Paola Marolo and Federico Ricardi. (https://www.frontiersin.org/articles/10.3389/fmed.2021.710329/full)
5. Introduction to Artificial Intelligence by Rose Velazquez (https://builtin.com/artificial-intelligence)
6. Artificial Intelligence: How is It Changing Medical Sciences and Its Future? by Kanadpriya Basu, Ritwik Sinha, Aihui Ong and Treena Basu (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7640807)
7. Official documentations of Pandas, Numpy, Tensorflow, matplotlib, jupytet, Python
8. Intuition of Adam Optimizer (https://www.geeksforgeeks.org/intuition-of-adam-optimizer/#:~:text=Adam%20optimizer%20involves%20a%20combination,minima%20in%20a%20faster%20pace.)