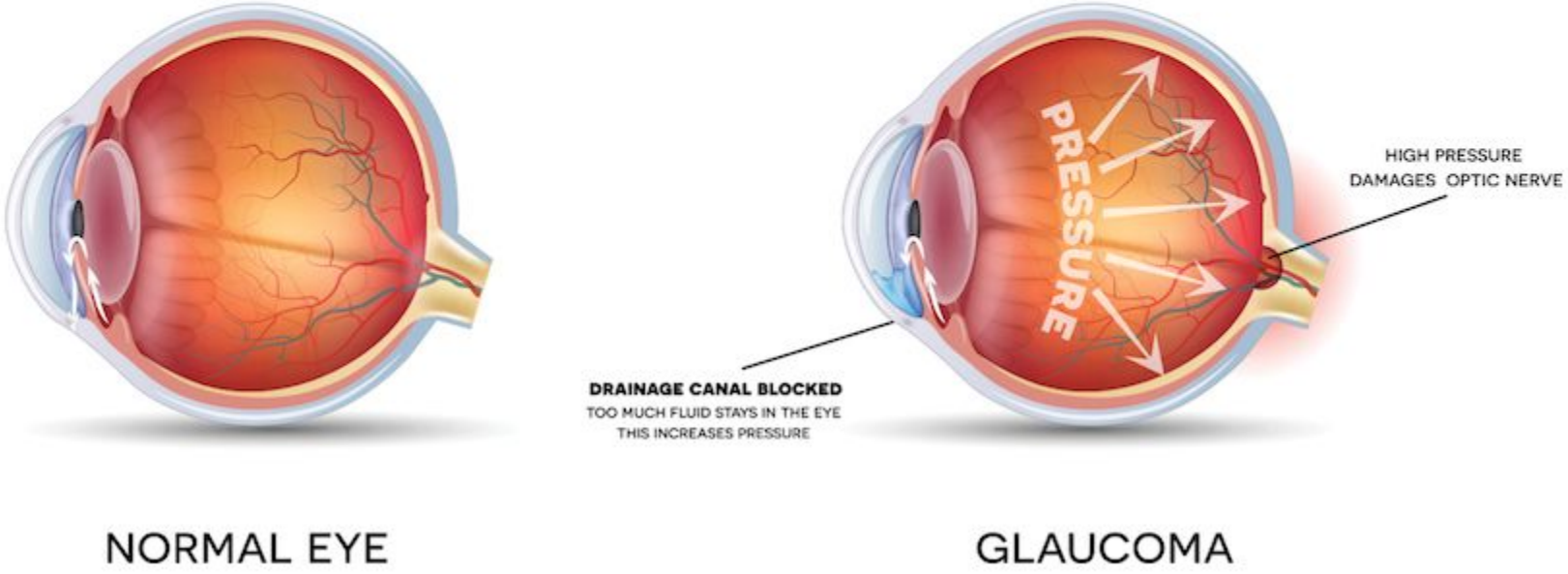
GLAUCOMA DETECTION USING CONVOLUTIONAL NEURAL NETWORKS(CNN)

SDG 3 - Good Health & Well-being

Disease detection with deep learning using medical imaging data is promising to provide affordable disease screening to alleviate societal disease burden and reduce health disparities between different demographic identity groups, which can be deployed in primary care and pharmacies without needing the subjects to visit the more expensive and busy specialty clinics

Glaucoma

An eye condition of increase pressure within the eyeball which causes loss of vision. It is characterized by increased intraocular pressure inside the eye leading to changes in the optic disc and optic nerve.



Problem Statement

- According to the World Health Organization (WHO), globally, approximately 217 million people have moderate to severe vision impairment. Glaucoma is one of the leading causes of irreversible blindness in people over 40 years old.

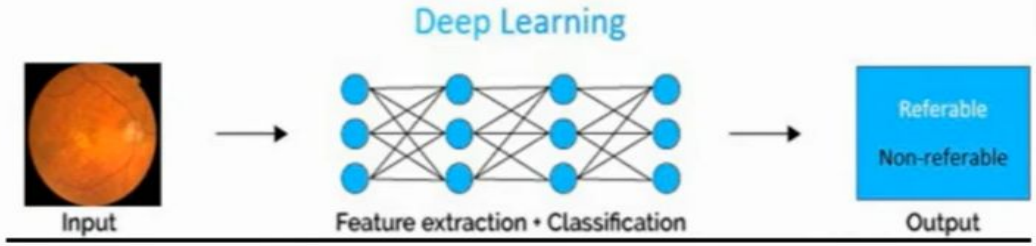
- A significant number of diseases disproportionately impact racial and ethnic minorities as well as socioeconomically disadvantaged demographics. Glaucoma in particular has been more severe in Black and Hispanic demographic.

Existing System for Detection

- Glaucoma is currently diagnosed using several methods, including a dilated eye exam to inspect the optic nerve for damage, which can appear as cupping. Additionally, an ocular pressure test (tonometry) measures the pressure inside the eye, as elevated pressure can indicate glaucoma. These assessments help ophthalmologist(eye doctor) determine the presence and severity of the disease.
- It does not reveal its symptoms until later stage. Hence, regular screening of the patients is required to identify the disease, thus demanding high labor, time and expertise.

Proposed System

- We propose a system that leverages deep learning to find patterns in labeled RNFLT (Retinal Nerve Fiber Layer Thickness) images in order to detect whether or not an individual has glaucoma. (Predictive data analytics)

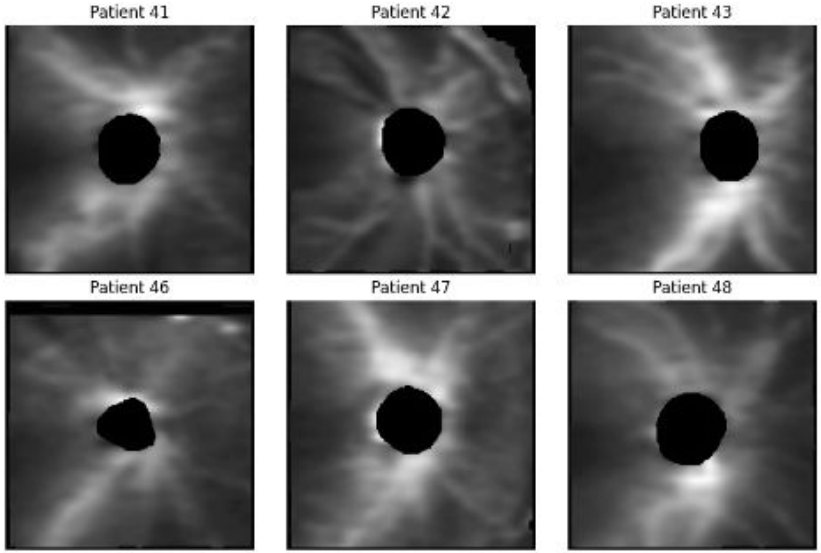


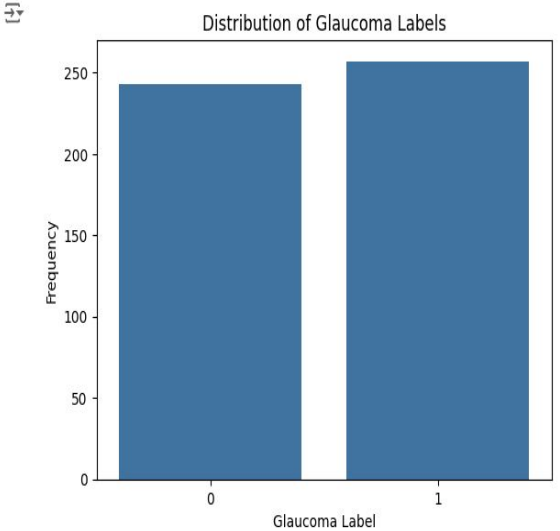
Using deep learning and CNN for training image dataset of gluacoma

The accuracy of CNN is superior in image recognition problem

Dataset

- We used a dataset of 500 images from Harvard Medical School. This datasets include RNFLT maps from 500 unique patient with and without glaucoma.





Data Preprocessing

- **Data Augmentation** - Random rotation / Zoom in-out / Random crop center
- **Cropped Image** - To remove artifact which are more likely to be present at edge of image
- **Image Repair and Reconstruction**
    - MedianBlur (filtering to remove noise from image)
    - Gaussian Blur (repairing image by introducing blurring filter)
    - Impainting (+mask) - reconstruct image by applying mask on dark spot and replacing them with surrounding area
- **Extract RGB color from gray scale**

Model Architecture

Our model is contains the following layers:

Convolutional Layers (Feature extraction):

- There are three convolutional layers with increasing filters (32, 64, 128), each followed by a max-pooling layer to reduce spatial dimensions.

Flattening (from 3D images to 1D vectors to prepare for dense layer)

- The output from the convolutional layers is flattened into a 1D vector.

Fully Connected Layers

- A dense layer with 128 units followed by a dropout layer to prevent overfitting.

Output Layer

- A dense layer with a sigmoid activation function for binary classification.

Model Training

- The model was trained using a process where it repeatedly learns from the training data over several iterations. The Final accuracy was 97%

Train Model

```
[ ]  # from tensorflow.keras.callbacks import ModelCheckpoint

     # checkpoint = ModelCheckpoint('model_checkpoint.h5', save_best_only=True, monitor='val_loss')

     history = model.fit(train_maps, train_labls,
                         epochs=9 ,
                         batch_size=32,
                         validation_data=(val_maps, val_labls),
                         verbose=1)
```

```
Epoch 1/9
63/63 [==============================] - 6s 71ms/step - loss: 14.4583 - accuracy: 0.6410 - val_loss: 0.5184 - val_accuracy: 0.7900
Epoch 2/9
63/63 [==============================] - 4s 64ms/step - loss: 0.4898 - accuracy: 0.7735 - val_loss: 0.4574 - val_accuracy: 0.7820
Epoch 3/9
63/63 [==============================] - 4s 63ms/step - loss: 0.4548 - accuracy: 0.7920 - val_loss: 0.4688 - val_accuracy: 0.7860
Epoch 4/9
63/63 [==============================] - 4s 66ms/step - loss: 0.4018 - accuracy: 0.8215 - val_loss: 0.4328 - val_accuracy: 0.7940
Epoch 5/9
63/63 [==============================] - 4s 65ms/step - loss: 0.3865 - accuracy: 0.8245 - val_loss: 0.4090 - val_accuracy: 0.8100
Epoch 6/9
63/63 [==============================] - 4s 64ms/step - loss: 0.3542 - accuracy: 0.8345 - val_loss: 0.4700 - val_accuracy: 0.7720
Epoch 7/9
63/63 [==============================] - 4s 66ms/step - loss: 0.3236 - accuracy: 0.8520 - val_loss: 0.4474 - val_accuracy: 0.8160
Epoch 8/9
63/63 [==============================] - 4s 67ms/step - loss: 0.3041 - accuracy: 0.8640 - val_loss: 0.4034 - val_accuracy: 0.8320
Epoch 9/9
63/63 [==============================] - 4s 65ms/step - loss: 0.2928 - accuracy: 0.8725 - val_loss: 0.4642 - val_accuracy: 0.8260
```

Fine Tuning

The Past Version of the model were overfitting. This is likely due to the data to epoch ratio.

- I started at 25 epochs, but I noticed that after around 9 epochs, the validation loss began to increase, while the train accuracy began to increase. This meant that the model was over familiarizing itself with the training dataset impairing the models ability to generalize.
- I modified the dataset I was using in a previous instance of my model. Initially, many images had dark spots, which limited the models ability to learn effectively. I leveraged Inpainting technique is used to reconstruct each image to fill in dark spots based on surroundings. My validation accuracy increased from around 7.7 to 8.0
- I included a third Convolution Layer in hopes of improving model accuracy, especially with regards to validation accuracy. I was able to bump validation accuracy of model to 8.2

Model Evaluation

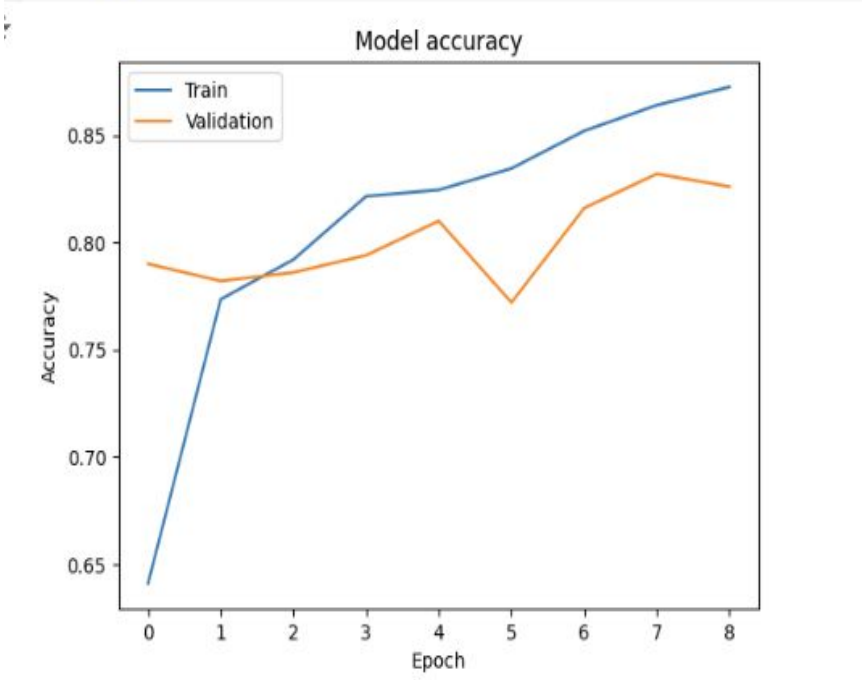
We could not find any other datasets that had RNFLT images so we segmented 32 samples for testing purposes:

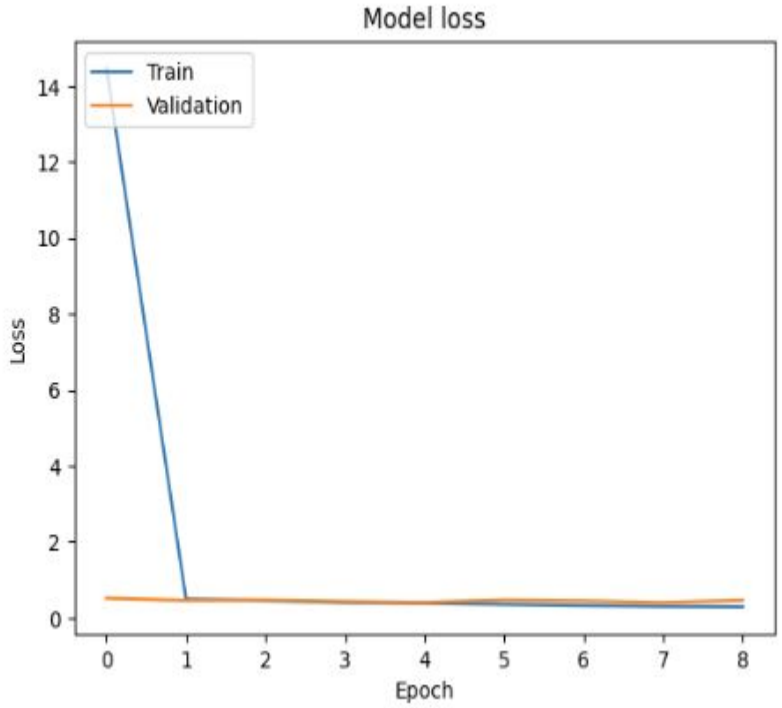
```
# Evaluate the restored model
loss, acc = new_model.evaluate(last_maps, last_labls, verbose=2)
print('Restored model, accuracy: {:5.2f}%'.format(100 * acc))

# Predict on the validation data (val_maps), not the labels
predictions = new_model.predict(val_maps).shape
print(predictions)
```

```
4/4 - 4s - loss: 0.1648 - accuracy: 0.9500 - 4s/epoch - 989ms/step
Restored model, accuracy: 95.00%
19/19 [==============================] - 25s 1s/step
(580, 1)
```

Model Evaluation





Initial Plan Of the Software Platform

- User-Friendly Interface - Designing a user interface for the screen tool that allows healthcare professionals and other individuals to capture eye image.

Software Requirements

- Operating System: Windows 7 or Higher
- Coding Language: Python
- FrontEnd: Streamlit
- BackEnd: TensorFlow and Keras

THANK YOU!