DEVELOPMENT OF AI-BASED ALGORITHM FOR RETINAL DISEASE SCREENING

DASWANI, SAHIL BHARAT

PROBLEM

"Globally, at least 2.2 billion people have a near or distance vision impairment. In at least 1 billion – or almost half – of these cases, vision impairment could have been prevented or has yet to be addressed. "

Causes of the Problem

- Limited accessibility to medical care
- Diagnosing Ocular Diseases is a time-consuming process

OBJECTIVES

1. To develop an AI image classifier to detect presence of different types of ocular diseases using fundus images with an accuracy of over 85%.
2. To experiment with different and combined convolutional neural network (CNN) architectures to achieve the best accuracy.
3. To develop a user-friendly web app that implements the AI model, which can be used by the public and medical professionals.

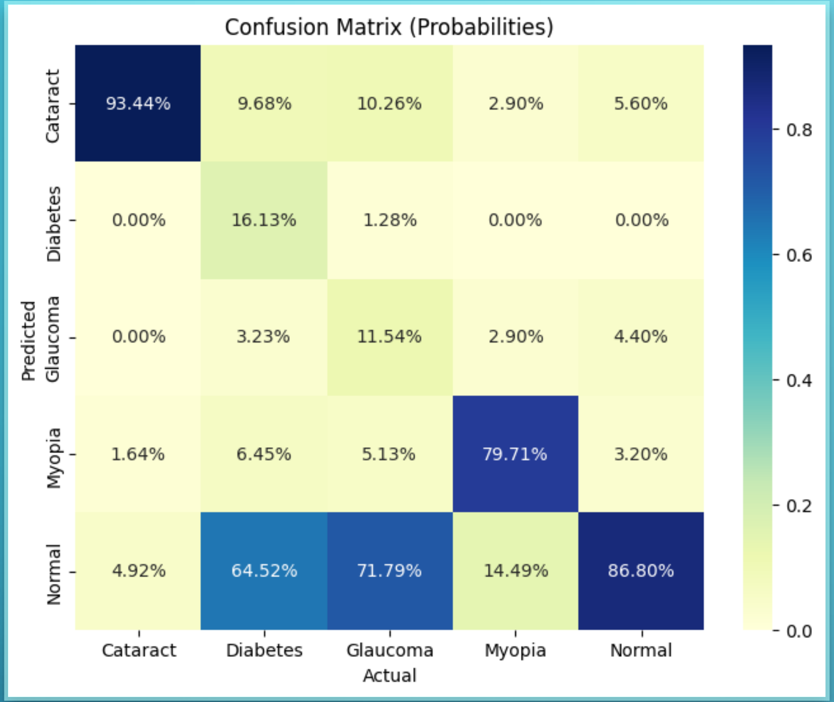
DEVELOPING THE IMAGE CLASSIFIER

- Trained The Model Using PyTorch
- Used Transfer learning for existing pre-trained Models: ResNet50, VGG16 and VIT

Steps:

1. Data analysis and data processing
2. Training the model
3. Evaluating the accuracy of the model
4. Optimize Hyperparameters
5. Repeat

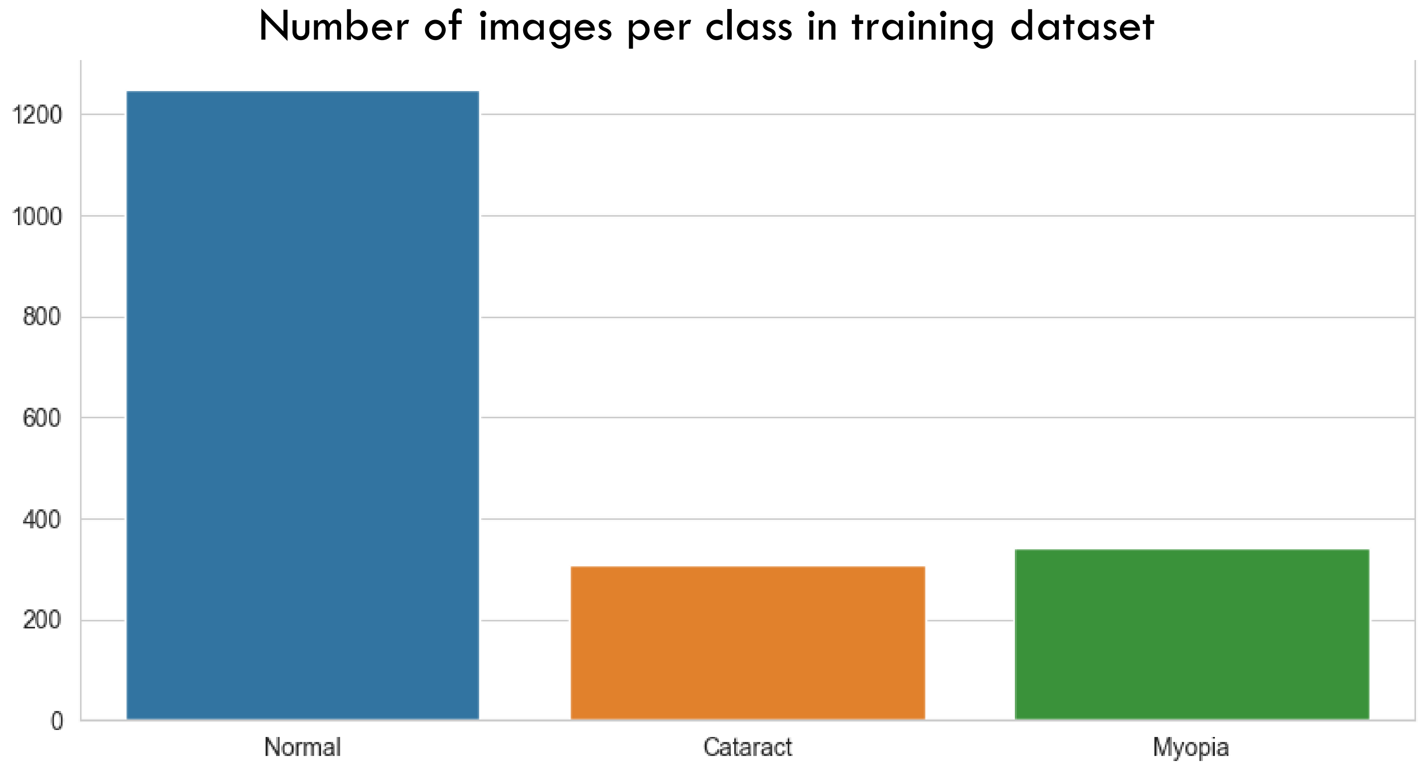
DIFFICULTIES OF CLASSIFYING MULTIPLE DISEASES



- Encountered difficulties classifying multiple diseases
- Diabetic Retinopathy and Glaucoma had bad accuracy
- Decided to reduce the number of classes to Normal, Cataract and Myopia

SKEWED DATASET

- The imbalance of data in each class caused a lot of bias
- Built an image augmenter function to augment images in each minority class.
- The augmenter creates new images by randomly flipping, rotating and adding blur to existing images.



TRANSFER LEARNING ALGORITHM

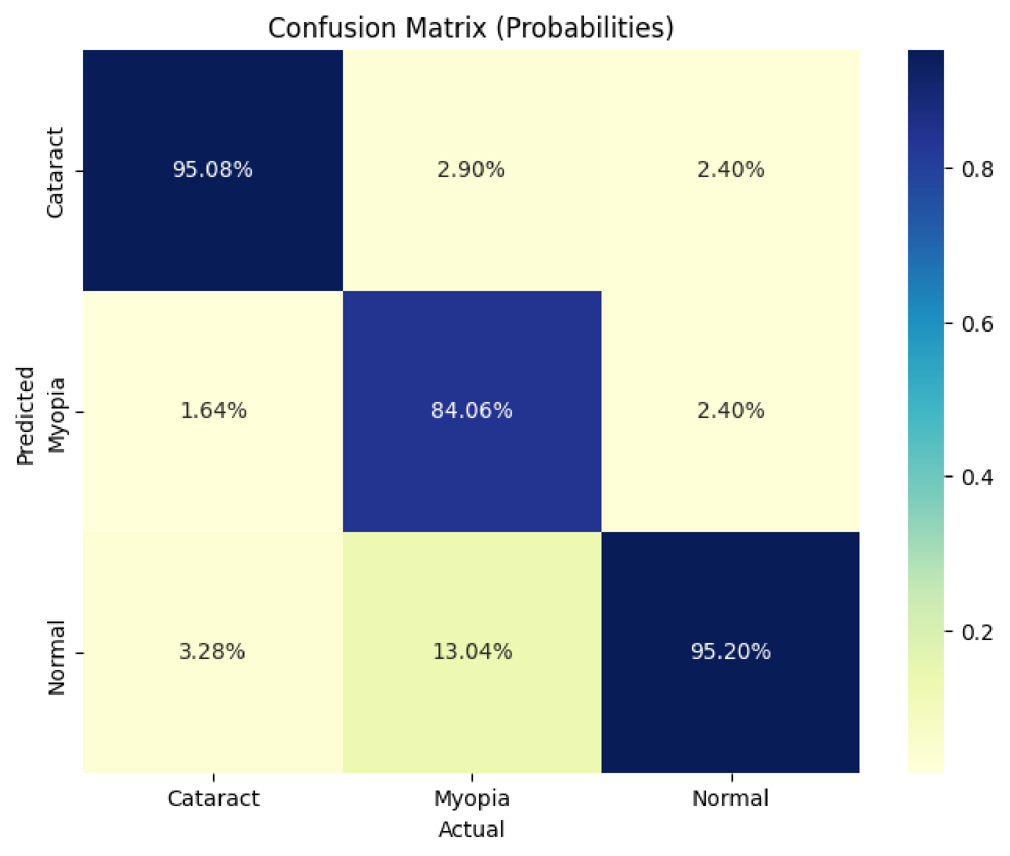
- The hyperparameters: learning rate, epochs and batch size were defined
- The cross-entropy loss function and Adam loss optimizer was defined
- During the training loop:
  - Input data and labels were moved to the GPU for acceleration
  - Reset the gradients of the optimizer
  - Forward pass: Feed the inputs to the model
  - Compute the loss
  - Backward pass: Compute gradients with respect to the loss
  - Update the model parameters using the optimizer

TRANSFER LEARNING ALGORITHM

- Training loop validation:
  - Move the input data and labels to the GPU.
  - Perform a forward pass to obtain the predicted outputs.
  - Compute the validation loss.
  - Compute a confusion matrix to evaluate the per-class accuracies.
  - Accumulate the validation loss and compute the average class accuracy.
- The model with the highest average class accuracy was saved

MODEL EVALUATION

- Load the saved model
- Loops through all the images and predict the diagnosis of all the fundus images in the testing dataset.
- Generated a confusion matrix to display all the class accuracies



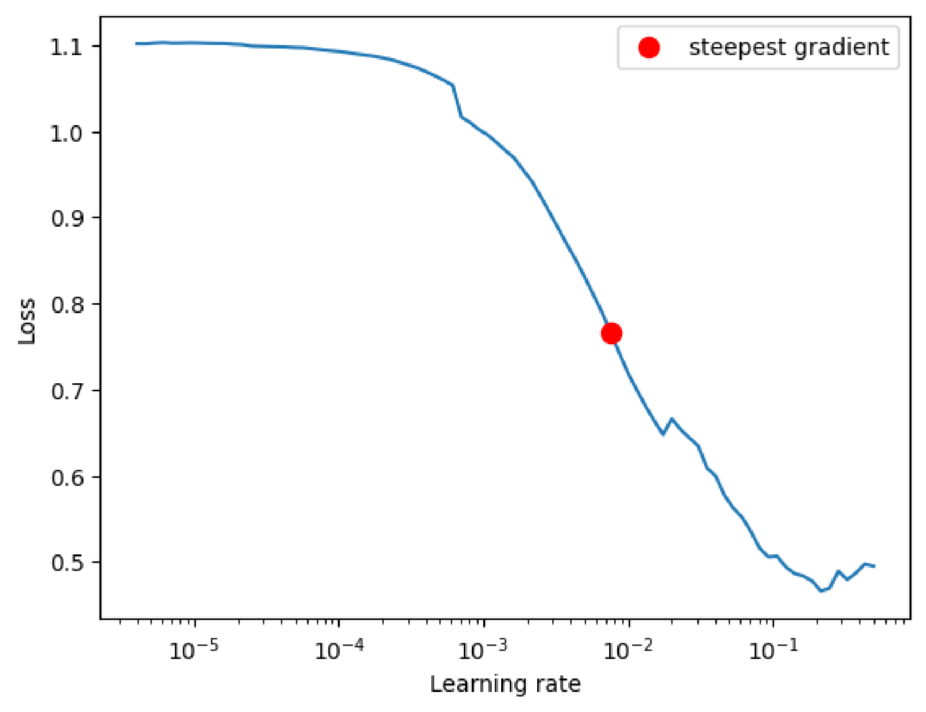
Confusion Matrix of ViT model

OPTIMISING THE HYPERPARMETERS

- The aim was to optimize batch size, epochs and learning rate
- Attempted to use a grid search to loop through a range of hyperparameter combinations and evaluate the accuracy of the model.
- This method was time consuming and was ineffective for finding the optimal values of epochs and learning rate.
- However, I figured out the optimal batch size for my GPU was 128 because any thing over that would overload the GPU and use a lot more memory swap.

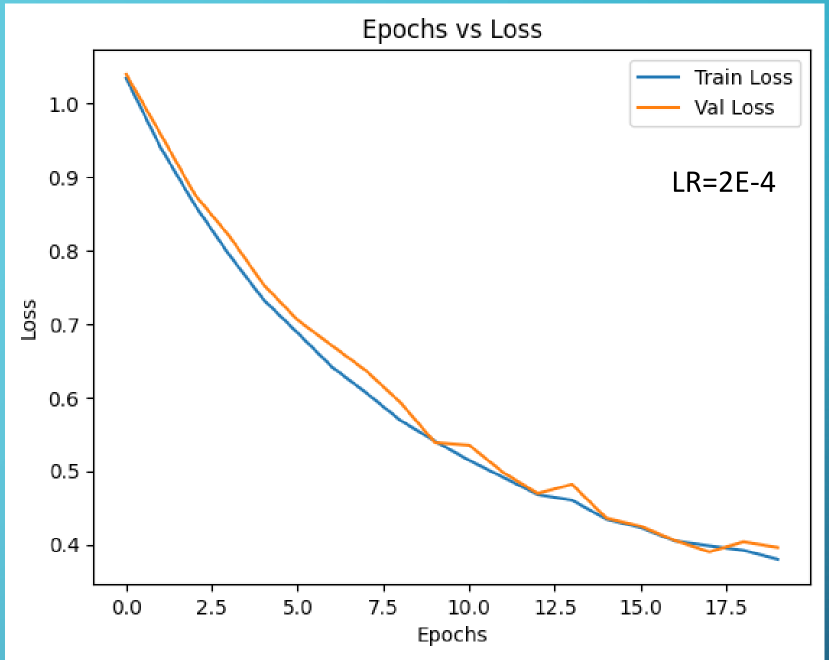
OPTIMISING THE HYPERPARMETERS

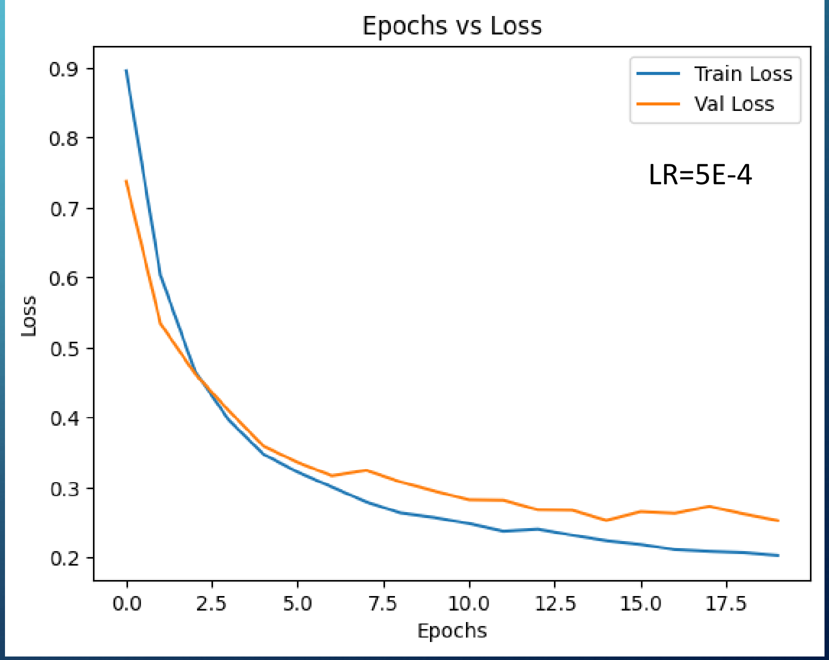
- The next attempt was to use a library called LR_Finder
- Plots the loss against the learning rate and finds points where the gradient is steepest to give the optimal learning rate.
- The learning rate given by the library was not always optimal.
- Used it as a starting point and used an incremental trial and error approach to optimize the learning rate



OPTIMISING THE HYPERPARMETERS

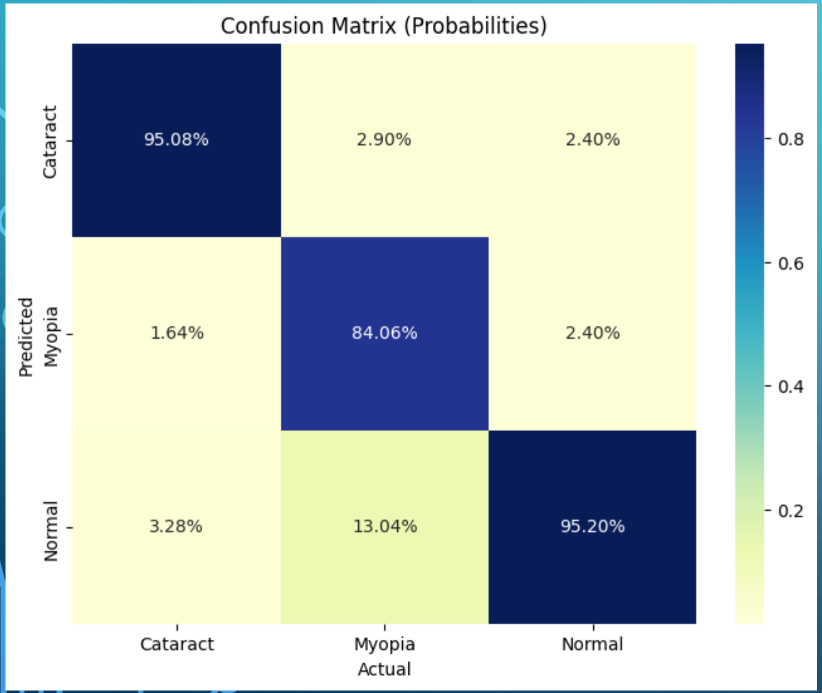
- Kept the epochs fixed and only adjusted the learning rate
- Realized that epochs is a tradeoff between training time and window to diverge at the optimal solution
- Kept reducing the learning rate incrementally using a trial-and-error approach until the loss converged.



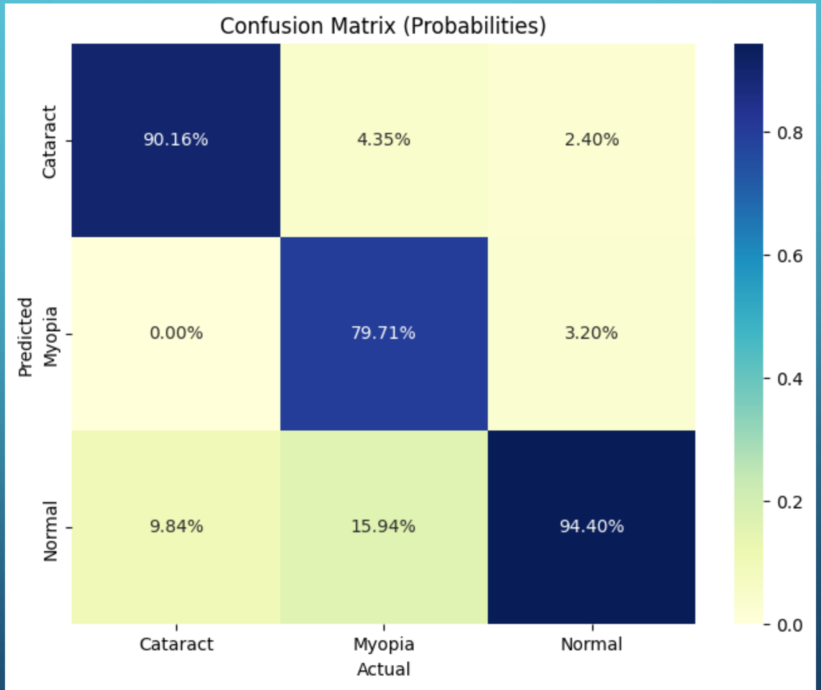


RESULTS

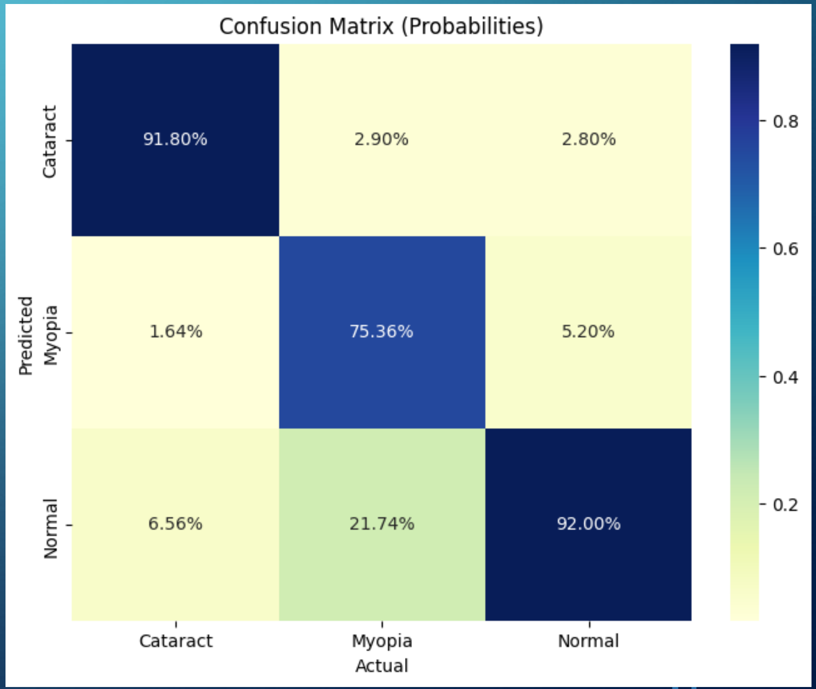
Model	Average Class Accuracy (%)
VGG16	86.39
ResNet50	88.09
VIT	91.45



VIT



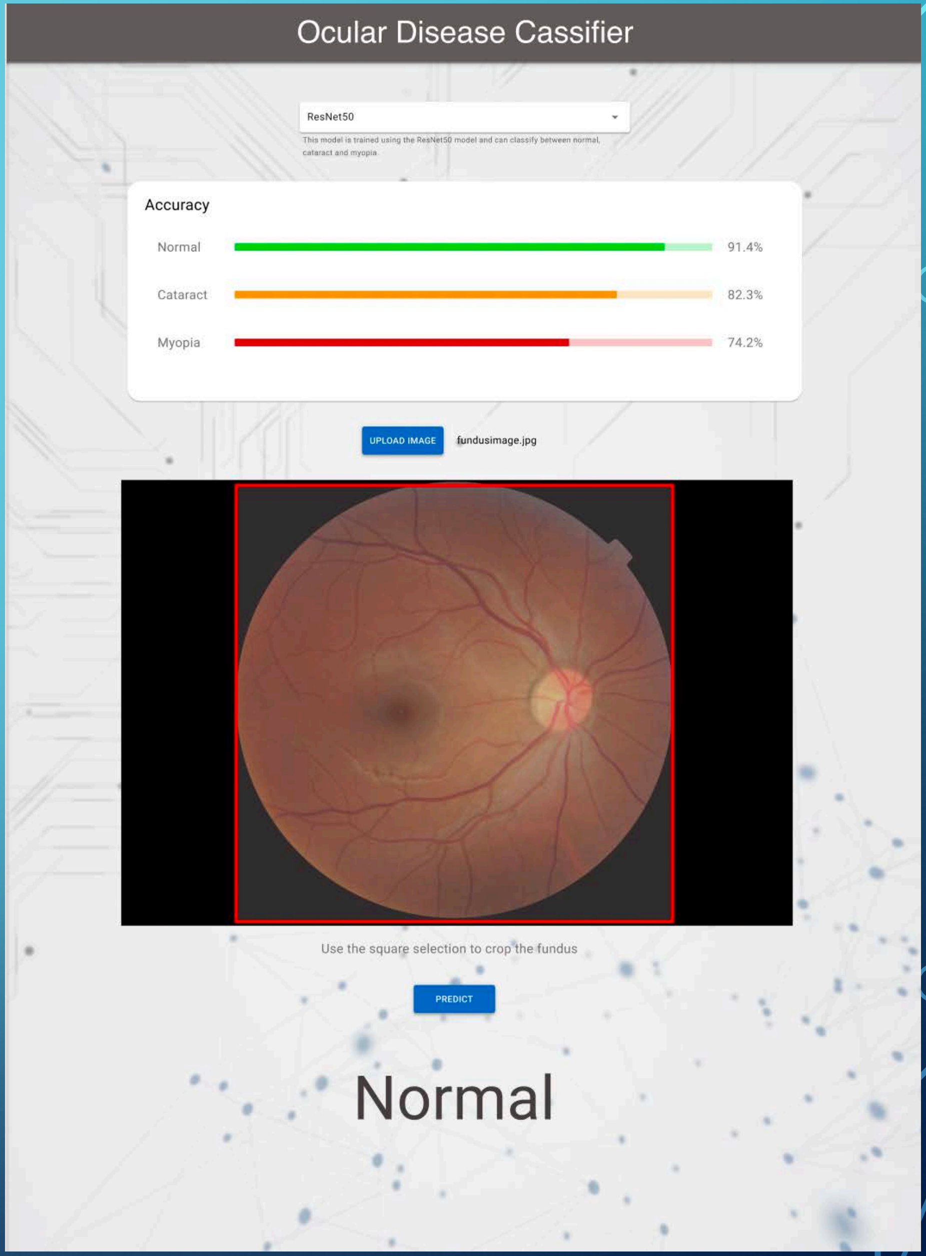
ResNet50



VGG16

DEVELOPING THE WEB APP

- Used Figma for the web design
- Used React.js and MaterialUI for the Client Side
- Used Flask to build the API and run the image classifier on the Server Side
- Collected all the inputted images in Firebase Storage for future improvements of the model.



Figma Design

DEMONSTRATION

CONCLUSION

- Achieved a 91.5% accuracy
- Tested out and compared the different CNN models and vision transformer.
- Successfully built a web application to run the models
- However, its limitation of only being able to classify between 3 classes makes it not useful enough yet to be implemented in the medical field
- Further improvements are needed to make it a viable product

FUTURE WORK

- Potential for semi-supervised learning using collected data
- Gather a much larger dataset of fundus images
- Explore Bayesian Optimization and Hyperparameter Importance Analysis for hyperparameter tuning
- Explore increasing the dataset size using stable diffusion