Detecting Retinal Detachment using images with Machine Learning

Ethan Lin
Leland High School
San Jose, California, USA

ABSTRACT

Retinal detachment occurs when your retina (a light-sensitive layer of tissue in the back of your eye) is pulled away from its normal position at the back of your eye. Symptoms include increased eye floaters, sudden flashes of light in vision, and dark shadows in the middle or on the sides of vision. It can eventually lead to blindness. Therefore, it is important that we detect it early on and treat it. This research is on how AI and machine learning can be used to detect retinal detachment using images. Before building my models, I did data pre-processing on retinal images data from four different hospitals. Then I built several different machine learning models, including Logistic Regression, Ridge Classifier, Decision Tree Classifier, K-nearest neighbor, Random Forest Classifier, SVM, MLP Classifier, Neural Networks, and Transfer Learning. I then assessed their accuracy using certain metrics. After building the models, I calculated the accuracy score, precision score, recall score, F1 score, and built Confusion Matrices for all of the models. Our results showed a very high accuracy in all of them; they were all in the 90% to 99% range. Therefore, I can assume that Retinal Detachment can be diagnosed using Machine Learning methods. In summary, retinal detachment is a serious issue and needs to be detected and treated. Using our machine learning models, I achieved great results.

INTRODUCTION

Retinal detachment is an eye problem that happens when your retina (a light-sensitive layer of tissue in the back of your eye) is pulled away from its normal position at the back of your eye. Symptoms include increased new eye floaters (small dark spots or string looking lines that float across your vision), flashes of light in one eye or both eyes, and a dark shadow/curtain on the sides or in the middle of your vision which decreases your vision scope. Retinal detachment is a medical emergency, therefore, it is important to be able to detect it and treat it. In this research, I am trying to see how machine learning models can be used to predict retinal detachment. This is a supervised learning problem, which means that the models will predict an output of a given input based on example inputs and outputs that I trained it with. In addition, this is also a classification problem, since the data I am working with is image data. Before I built our models, I first found the image data that I am using. The data I am using is from IEEE DataPort, and the name of the dataset is called Retinal Detachment Dataset. In this dataset, there is image data from four different hospitals, each image already sorted into "Retinal detachment" or "Not Retinal detachment". For pre preprocessing, I first created two lists, RD (retinal detachment) and Non RD (Not retinal detachment). I sorted all the images into these two lists, resized them, transformed the list into numpy arrays, and created labels for them. Lastly, I normalized the images and converted them to gray scale. After preprocessing, I splitted the data into training and testing sets, then developed the basic models: Logistic regression, Ridge Classifier, Decision Tree Classifier, K-nearest neighbor, Random Forest Classifier, SVM, and MLP Classifier. Then I moved on to advanced models, Neural Networks and Transfer Learning. I trained all of the models with the image dataset that I splitted into training and testing sets. For each of the models, I calculated their accuracy score, precision score, recall score, and F1 score. Based on these scores, I can see how well our model performed.

BACKGROUND

The idea of this research is to use machine learning models to identify and classify retinal detachments in images. Some models that are relevant to this experiment are Logistic Regression, Ridge Classifier, Decision Trees, K-Nearest Neighbors, Random Forest Classifier, SVM, MLP Classifier, Convolutional Neural Networks, and Transfer Learning. Current research has also used convolutional neural network architecture Inception v3 to train, validate, and test deep learning models to predict the anatomical outcome of RRD surgery. They also used the area under the curve (AUC), sensitivity, and specificity for predicting the outcome. This previous research model was able to achieve an AUC of 0.94, with a corresponding sensitivity of 73.3% and a specificity of 96%. Another similar research was done by Gehad A. Saleh, Nihal M. Batouty, Sayed Haggag, Ahmed Elnakib, Fahmi Khalifa, Fatma Taher, Mohamed Abdelazim Mohamed, Rania Farag, Harpal Sandhu, Ashraf Sewelam, and Ayman El-Baz. In this research, they used models such as SVM, Random Forest, Neural networks, convolutional neural networks, and more. In addition, they also assessed Specificity, Recall, accuracy, F1-score, Precision, and Kappa. I did all of these metrics except for specificity and Kappa in my research. This approach is unique because they included more metrics than I did. Lastly, another research done by Hideharu Ohsugi, Hitoshi Tabuchi, Hiroki Enno & Naofumi Ishitobi also focused on the detection of retinal detachment. In their research, they also used CNN and SVM. However, their CNN was a lot more deep and had a lot more hidden layers, which is a benefit. They were able to get a sensitivity of 97.6%, and a recall of 96.5%. In general, all of the research done before has all included the models that I used, and all obtained some great results.

DATASET

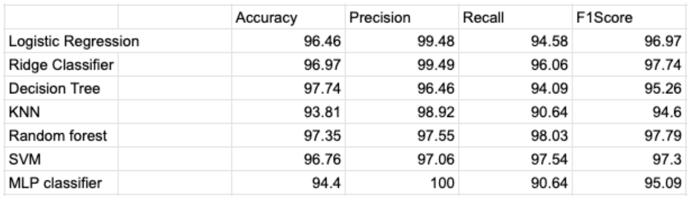
The data I used is from IEEE DataPort, and the name of the dataset is called Retinal Detachment Dataset. The database that I used for my research consists of both non-retinal detachment and rhegmatogenous retinal detachment images. These images were collected from four different hospitals: Silchar medical college and hospital, Aravind eye hospital, LV prasad eye hospital, and Medanta. There are a total of 1693 images in which 1017 images are retinal detachments and 676 images were non retinal detachments. It is great that this dataset already sorted the two into different groups, so it makes my data pre processing easier because I already know which image is retinal detachment and which ones are not. The original dimension of these images was fixed to the pixel size of 1000x1000x3. Once I imported the data, I resized the images to 128x128 and created two lists, RD (retinal detachment) and Non RD (no retinal detachment) and sorted each one correspondingly. Next, I transformed the lists into numpy arrays, because this is how I am going to work with the images. After transforming them into numpy arrays, I combined the two arrays into one big numpy array. Then I created labels for the images, with 0 being Non RD and 1 being RD. After the labels are created, I can view any of the images I want by using the plt.imshow function. Finally I normalized the images and converted them to grayscale. Normalizing an image means changing the range of the pixel values, and converting to grayscale means converting the image colors to different shades of gray. After I processed the images, I split them into training and testing groups so I can use them to train the machine learning models. I split the data by using the train_test_split function. After spitting, 1354 of the 1693 images were classified as the train set and 339 of the 1693 images were classified as the test set. Essentially, approximately 80% of the images were used as train data and 20% were used as test data.

METHODOLOGIES/MODELS

To solve this problem, I approached it with several machine learning models. And calculating their accuracies. However, before I created the models, I had to process my dataset first. After preprocessing my dataset, I splitted them into training and testing data by using the train_test_split function and passing in the parameters. In addition, I also gray scaled the images because it makes the data less complicated for the models. The basic models include Logistic Regression, Ridge Classifier, Decision Trees, K- Nearest Neighbors, Random Forest Classifier, SVM, and MLP Classifier. The advanced models are convolutional neural networks and Transfer Learning. Logistic Regression estimates the probability of an event occurring, in our case, it will be either Retinal Detachment or No Retinal Detachment. Ridge Classifier first converts the target values into {-1, 1} and then treats the problem as a regression task. Decision trees classify things by creating a decision tree, branching into more sections. K-Nearest Neighbors creates regions and boundaries, and uses proximity to make classifications. Random Forest Classifier combines the output of multiple decision trees to reach a single result. SVM, or support vector machine, is especially good for binary classifications where the classification leads to two groups. SVM maps data into a high dimensional feature space so that data points can be categorized. To make things more efficient, I used a for loop to train and test three of my baseline models: K-Nearest Neighbors Classifier, SVC, and Random Forest Classifier. After the baseline models, I started on neural networks. First, MLP Classifier is a kind of neural network where it updates the parameters each time it is run. Next, Convolutional neural network is a model that has layers, which detects different features in the imputed image. It is good at identifying patterns by using convolutional layers. For example, Rectified Linear Unit (ReLU) is a nonlinear activation function. It is important to have a nonlinear activation function because without it, no matter how deep the model is, it would always be equivalent to a linear model. The final layer is the softmax classifier. The softmax function gives a probability vector that sums to one. In a CNN, a filter or kernel is applied to produce an output that gets progressively better after each layer. Kernel is the pattern that we are looking for, it is represented as numbers. However, since neural networks cannot handle categorical data directly, I had to use one hot encoding to convert the categorical data to numerical format. One hot encoding works by representing each category by using either 0 or 1 that indicates if the data belonged to a category or not. Lastly, I used Transfer Learning. Transfer learning is when the machine uses already known knowledge to help generalization easier in the future. I used a few convolutional neural network architectures already made for vision tasks such as VGG16. VGG19, ResNet50, etc. After building the models, I calculated the accuracy score, precision score, recall score, and F1 score for the basic models. To get better results, I hypertuned some of the models. Hypertuning is changing the hyperparameters of each model to try to get better results. For example, changing the numbers of neighbors in the KNN Classifier. I used a for loop to test different numbers of neighbors because it is more efficient than testing different numbers of neighbors individually. For CNN, before I ran the neural network, I reduced the size of the output and made everything one dimensional (flattening). This makes it less complicated. In addition to calculating its accuracy, I also plotted the validation and training on a graph to check for overfitting. Overfitting is when a model is trained too much on training data that it performs well on the training data but will do worse when it is given actual data. Just like I hypertuned the baseline models to get better results, I can do the same with Convolutional Neural Networks even though its set up is different. For CNN, I could add or remove hidden layers, or change the number of epochs. Epoch is each time the training data is passed through an algorithm, in this case, the neural network.

RESULTS AND DISCUSSION

For each of the baseline models and the MLP classifier, I calculated their accuracy score, precision score, recall score, and F1 score. In addition, I also ran a confusion matrix for these models. A confusion matrix represents the prediction summary in matrix form. It has four sections: True Negative, False Positive, False Negative, and True Positive. The section True Negative shows the situation where the prediction is negative and it is actually negative. Similarly, a True Positive represents situations where the predicted value is positive and the data is actually positive. False Positive is when the predicted value is positive but it is actually false. Lastly, False Negative is when the prediction is negative but it is actually positive. Having a confusion matrix is helpful because it allows me to see the models' performance and what went wrong.
Table summary of results:

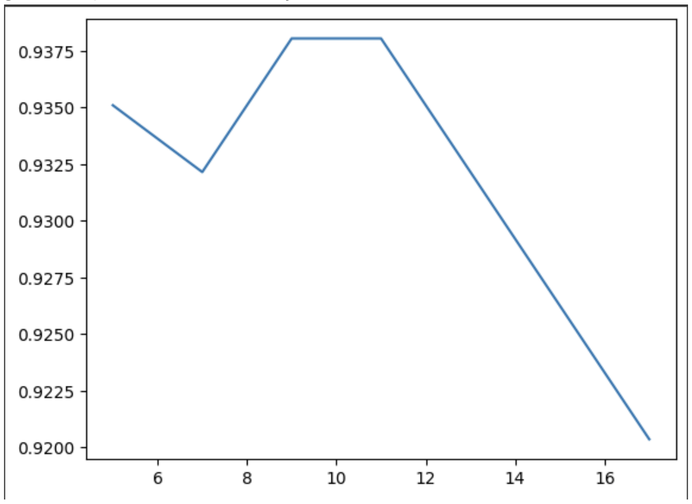


	Accuracy	Precision	Recall	F1Score
Logistic Regression	96.46	99.48	94.58	96.97
Ridge Classifier	96.97	99.49	96.06	97.74
Decision Tree	97.74	96.46	94.09	95.26
KNN	93.81	98.92	90.64	94.6
Random forest	97.35	97.55	98.03	97.79
SVM	96.76	97.06	97.54	97.3
MLP classifier	94.4	100	90.64	95.09

For the first baseline model Logistic Regression, I obtained an accuracy score of 96.46%, a precision score of 99.48%, a recall score of 94.58%, and a F1 score of 96.97%. For its confusion matrix, there were 135 True Negatives, 1 False Positive, 11 False Negatives, and 192 True Positives. For the second baseline model Ridge Classifier, I obtained an accuracy score of 96.97%, a precision score of 99.49%, a recall score of 96.06%, and a F1 score of 97.74%. For its confusion matrix, there were 135 True Negatives, 1 False Positive, 8 False Negatives, and 195 True Positives. For the third baseline model Decision Tree Classifier, I obtained an accuracy score of 97.74%, a precision score of 96.46%, a recall score of 94.09%, and a F1 score of 95.26%. For its confusion matrix, there were 129 True Negatives, 7 False Positives, 11 False Negatives, and 192 True Positives. For the fourth baseline model K-Nearest Neighbors Classifier, I obtained an accuracy score of 93.81%, a precision score of 98.92%, a recall score of 90.64%, and a F1 score of 94.6%. For its confusion matrix, there were 134 True Negatives, 2 False Positives, 19 False Negatives, and 184 True Positives. For the fifth baseline model Random Forest Classifier, I obtained an accuracy score of 97.35%, a precision score of 97.55%, a recall score of 98.03%, and a F1 score of 97.79%. For its confusion matrix, there were 132 True Negatives, 4 False Positives, 4 False Negatives, and 199 True Positives. For the sixth baseline model SVC, I obtained an accuracy score of 96.76%, a precision score of 97.06%, a recall score of 97.54%, and a F1 score of 97.3%. For its confusion matrix, there were 130 True Negatives, 6 False Positives, 5 False Negatives, and 198 True Positives.

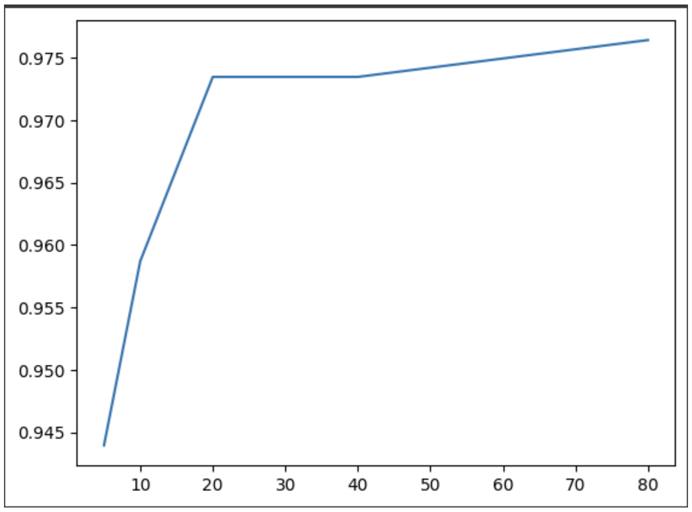
After these baseline models, I moved on to neural networks. The first one is the MLP classifier. For the MLP classifier, I obtained an accuracy score of 94.4%, a precision score of 100%, a recall score of 90.64%, and a F1 score of 95.09%. For its confusion matrix, there were 136 True Negatives, 0 False Positives, 6 False Negatives, and 197 True Positives. The next neural network I built is a Convolutional Neural Network. For 15 epochs with a convolutional layer, an activation function, pooling, flattening, and a relu function and a softmax function, the highest accuracy I obtained was 97%. I graphed the number of epochs against the accuracy and found that there is a little bit of overfitting.

I also hypertuned the KNN Classifier and the Random Forest Classifier. For the KNN classifier, I changed the number of neighbors to 5, 7, 9, 11, and 17. For 5 neighbors, I obtained an accuracy score of 93.51%, a precision score of 97.38%, a recall score of 91.63%, and a F1 score of 94.42%. For 7 neighbors, I obtained an accuracy score of 93.22%, a precision score of 96.88%, a recall score of 91.63%, and a F1 score of 94.18%. For 9 neighbors, I obtained an accuracy score of 93.81%, a precision score of 95.96%, a recall score of 93.6%, and a F1 score of 94.76%. For 11 neighbors, I obtained an accuracy score of 93.81%, a precision score of 96.43%, a recall score of 93.1%, and a F1 score of 94.74%. For 17 neighbors, I obtained an accuracy score of 92.04%, a precision score of 95.36%, a recall score of 91.13%, and a F1 score of 93.2%. I plotted the number of neighbors on the x axis and the accuracy on the y axis and found that in general, as the number of neighbors goes up, the accuracy decreases.



For the Random Forest Classifier, I changed the n_estimators (the number of trees in the forest) to 5, 10, 20, 40, and 80. For n_estimators = 5, I got an accuracy score of 94.4%, a precision score of 97.42%, a recall score of 93.1%, and a F1 score of

95.21%. For n_estimators = 10, I got an accuracy score of 95.87%, a precision score of 98.46%, a recall score of 94.58%, and a F1 score of 96.48%. For n_estimators = 20, I got an accuracy score of 97.35%, a precision score of 97.09%, a recall score of 98.52%, and a F1 score of 97.8%. For n_estimators = 40, I got an accuracy score of 97.35%, a precision score of 97.09%, a recall score of 98.52%, and a F1 score of 97.8%. For n_estimators = 80, I got an accuracy score of 97.64%, a precision score of 98.03%, a recall score of 98.03%, and a F1 score of 98.03%. I plotted the number of trees against the accuracy and found that as the number trees increase, the accuracy also increases.



Despite the great results, they were unexpectedly high. A potential reason for the high result is that the images in my dataset are too similar. This can cause the models to recognize patterns too easily. Another issue is that since I have so much data and code, Google Collab was having difficulty remembering the previous code. As a result, I had to keep going in and retype the previous codes.

CONCLUSION

Retinal detachment is an issue that is serious and could lead to the loss of sight. Therefore, it is important to be able to recognize them early on. As a result, I tried to develop machine learning models to solve this problem. First, I found a dataset that contained images of retinal detachments and non retinal detachments from hospitals. After I preprocessed my data, I splitted them into training and testing groups. With this training and testing data, I trained them with the models that I built: Logistic Regression, Ridge Classifier, Decision Tree Classifier, KNN Classifier, Random Forest Classifier, SVC, and neural networks such as MLP Classifier and Convolutional Neural Network. The models performed really well and I obtained unexpectedly high accuracy. Even though this is a good thing, I have to take into account possible reasons for that. For example, the images from the dataset could have been so similar to each other that it is too easy for the models to recognize patterns. Next steps that I could take are trying out different models, changing the epochs and hidden layers in the CNN, and hypertuning even more models to obtain a better result. Lastly, a possible extension of my research is to try to use these models on similar problems that involve image recognition.

ACKNOWLEDGMENTS

I would like to thank R Murugan and Sonal Yadav from the National Institute of Technology Silchar for providing the dataset that I used.

I would also like to thank Samar Abu Hegly for guiding me through this research and providing help when needed.

REFERENCES

- Ohsugi, H., Tabuchi, H., Enno, H., & Ishitobi, N. (2017, August 25). *Accuracy of deep learning, a machine-learning technology, using ultra–wide-field fundus ophthalmoscopy for detecting Rhegmatogenous Retinal Detachment*. Nature News. https://www.nature.com/articles/s41598-017-09891-x#Sec4
- Saleh, G. A., Batouty, N. M., Haggag, S., Elnakib, A., Khalifa, F., Taher, F., Mohamed, M. A., Farag, R., Sandhu, H., Sewelam, A., & El-Baz, A. (2022a, August 4). *The role of medical image modalities and AI in the early detection, diagnosis and grading of retinal diseases: A survey*. MDPI. https://www.mdpi.com/2306-5354/9/8/366

- Fung, T. H. M., John, N. C. R. A., Guillemaut, J.-Y., Yorston, D., Frohlich, D., Steel, D. H. W., & Williamson, T. H. (2022, October 28). *Artificial intelligence using deep learning to predict the anatomical outcome of Rhegmatogenous retinal detachment surgery: A pilot study - graefe's archive for clinical and experimental ophthalmology*. SpringerLink. https://link.springer.com/article/10.1007/s00417-022-05884-3
- NIH. (2022, April 21). *Retinal detachment*. National Eye Institute. https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/retinal-detachment