ECE 8803 Term Project: Comparison between Four Classification Methods for DRSS Severity Classification on OCT Images

Tawhid Khan, Syed Anas Hussain

School of Electrical and
Computer Engineering, Georgia
Institute of Technology, Atlanta,
GA, 30332-0250.
tkhan60@gatech.edu
shussain81@gatech.edu

I. Introduction

This project involved applying four different classification techniques on the OLIVES OCT image dataset to conduct Diabetic Retinopathy Severity Scale (DRSS) classification. The first part involves applying the techniques that take only the OCT images as input samples and output the associated DRSS severity levels, which are either 0, 1, or 2. The techniques used are K-Nearest Neighbors (KNN), Multi-class Logistic Regression, Convolutional Neural Networks (CNN), and Random Forest. The second part involves a comprehensive evaluation of the performance of the different techniques using metrics such as accuracy, balanced accuracy, precision, recall, F1 score, true positive rate, and false positive rate. The Github Repository to the presentation is HERE

II. Classification Algorithms and Methodology

A. *K-Nearest Neighbors (KNN)*

The first method used is K-Nearest Neighbors (KNN), a non-parametric discriminative classifier. This method classifies new samples based on how similar it is to previous cases. KNN uses a distance metric to calculate the similarity between all other previous samples and the new sample. The new sample is assigned to the class label based on the majority class label from among its 'K' nearest neighbors. Fig 1. illustrates an example of a KNN method with K=3 and a dataset containing two features. The Euclidean distance between the new sample and all other samples is calculated. The majority of its 3 nearest neighbors are blue, so this new sample is assigned blue label.

When K is too small there is a risk of overfitting because it will be sensitive to noise and outliers in data. The model will fail to generalize and capture noise in data instead of underlying patterns.

We used scikit-learn's KNN classifier for our experiment [1]. The input to the model during training was a 224x224 pixel greyscale OCT image and its DRSS level for all the images in the training dataset.

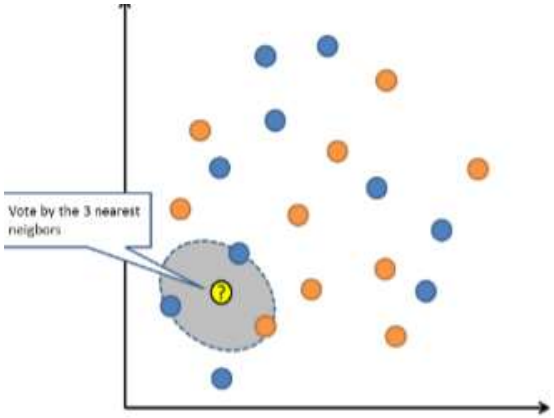

Fig 1. KNN with K=3

B. *Multi-class Logistic Regression*

Multi-class Logistic Regression is the second method we used to classify the OCT images. This method is a parametric discriminative classifier. It directly estimates the posterior probability for the 3 DRSS severity levels and then assigns the level with the highest posterior probability.

We used the Multi-class Logistic Regression model from the scikit-learn library [1]. The Stochastic Average Gradient (SAG) solver was used for optimization. SAG has many advantages, such as converging fast for large datasets such as OCT images. It is also memory efficient and can effectively handle multiclass classification problems with softmax activation functions.

This method first initialized the hyperparameters such as regularization parameter, maximum number of iterations, and the convergence tolerance. It then computed the weighted sum of inputs by multiplying

feature matrix with initial parameters. It then fed the output to the softmax activation function. Finally, it used SAG to solve cross-entropy loss and update the model parameters.

After repeating the above process until it converged or reached maximum iterations, the model predicted on the test data and selected the highest-class posterior probability as its prediction.

C. Convolutional Neural Network (CNN) - AlexNet

Convolutional Neural Network (CNN) is a type of neural network used widely for applications involving image classification. CNN typically comprises of multiple layers of 'neurons' that perform operations such as convolution and pooling going from one layer to another and extracting more and more detailed features that enable it to perform classification with a fairly high accuracy. The CNN architecture used in this project is AlexNet.

AlexNet is a deep convolutional neural network architecture that was introduced by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012 [2]. AlexNet achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge. Fig 2. depicts the architecture of AlexNet. It consists of five convolutional layers followed by three fully connected layers. The first layer helps to extract basic features from the image, such as edges and colors. The layers that follow learn increasingly complex features. The pooling layers downsample the feature maps, reducing the network's computational complexity. Finally, the three fully-connected layers FC6, FC7, and FC8 with 4096, 4096, and 1000 neurons help flatten the feature maps and classify the image into one of the 1000 ImageNet categories. The fully connected layers use a combination of dropout and ReLU activations to prevent overfitting and improve performance.

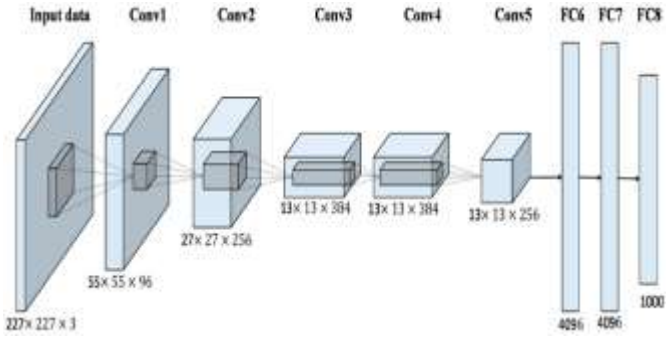

Fig 2. Architecture of the AlexNet CNN

In the context of our project, the AlexNet architecture had to modified slightly to accommodate for the size of the input images (224 x 224 pixels) as well as the number of class labels corresponding to the three different DRSS levels. A pre-processing stage was run on the input images to transform them to (64 x 64 pixels). This was primarily done to speed up the training time without compromising significantly on the feature space. Furthermore, since there are only three classes, the dimensionality of the last FC layer had to be reduced from 4096 to 3 instead of 4096 into 1000, as was used for the 1000 classes in ImageNet.

D. Random Forest

Random Forest is an ensemble learning algorithm that can be used for classification and regression. It works by building a multitude of decision trees during training and finally outputting the class that is the mode of the classes, in case of a classification task, or mean prediction of the individual trees, in the case of a regression problem [3]. Random forests correct for decision trees' habit of overfitting to their training set.

The Random Forest algorithm works by constructing multiple decision trees, each with a random subset of the features and a random subset of the training instances. Each of these decision trees is trained independently and makes a classification decision based on the majority vote of its constituent trees. The final classification decision is based on the majority vote of all the trees in the forest. Random Forest has several hyperparameters that can be tuned for performance optimization. Some of these include 'n estimators' - the number of decision trees in the forest, and 'max_depth' – the maximum depth of each decision tree. In general, as the number of trees, is increased, the performance of the model is better, but this comes at a cost of increased computational complexity. On the other hand, a deeper tree model determined by the 'max_depth' parameter can model more complex relationships in the data, but this can also lead to overfitting.

Because of its simple operating mechanism but effective performance, the Random Forest algorithm was selected as the fourth algorithm for OCT image classification.

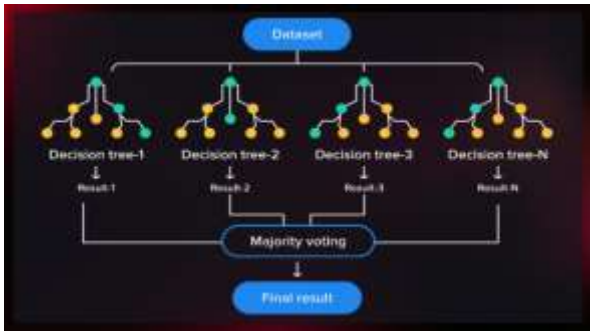

Fig 3. Random Forest algorithm based on Decision Trees

III. Experimental Results & Analysis

A. K-Nearest Neighbors (KNN)

Table 1. shows performance results for an experiment run with 4 different K values ranging from 1 to 25. All models had similar accuracies, either 39% or 40% and a balanced accuracy score of 33 to 34%. The accuracy gives the percentage of predicted labels on the test dataset that matched the actual labels. Balanced accuracy is a more appropriate metric to use than accuracy because the OCT train and test dataset contain imbalanced classes.

Precision, Recall, and F1 metrics were calculated using false positive, false negatives, true positive, and false negative values for each class. The weighted option was chosen, so metrics for each of the 3 classes were calculated and then a weighted average was calculated based on the number of samples in the class.

The precision metric measures how many of the predicted positives were true positives. Precision for all 4 models was 0.38. This meant that there was high rate of false positives since only 38% of positives were true positives. Recall measures true positive predictions out of all labeled positives. Recall was 39% for first 3 models and 40% for model with K=25. This means the model is at best estimating 40% of actual positive cases.

F1 score measures the balance between precision and recall but this was also 38-39% for all 4 models. Because, balanced accuracy, precision, recall, and F1 score did not vary greatly even when K increased, increasing K value did not have effect on improving the performance of the model.

On the other hand, as the value of K increased, the true positive rate decreased from 81% to 74% and false positive rate increased from 71% to 92%, meaning the model produced more false positives and fewer true negatives. This is perhaps because the dataset is imbalanced. As we increased the number of neighbors there was higher bias to the majority class leading to a lower true positive rate for minority classes and decreasing overall true positive rate.

Table. 1: KNN Performance with varying K values

	K = 1	**K = 5**	**K = 10**	**K = 25**
Accuracy	0.39	0.39	0.39	0.40
Balanced Accuracy	0.34	0.34	0.33	0.33
Precision	0.38	0.38	0.38	0.38
Recall	0.39	0.39	0.39	0.40
F1 Score	0.39	0.39	0.38	0.39
True Positive Rate	0.81	0.77	0.73	0.74
False Positive Rate	0.71	0.83	0.82	0.92

B. Multi-class Logistic Regression

For Multi-class Logistic Regression, the maximum number of iterations was set as 1000 and a SAG solver was used to run the logistic regression model import from the scikit-learn library [1]. The resulting performance metrics are shown in Table 2. The accuracy was 38% and the balanced accuracy was 31% which was lower than every K-NN model's accuracy and balanced accuracy scores. Other metrics such as precision, recall, and F1 score, were similar to the KNN metrics. The true positive and false positive rates were worse than the metrics from KNN. The false positive rate of 152% is more than twice the false positive rate of 71% from the KNN model with K=1. This is probably because logistic regression assumes a linear relationship between features and the target variable while KNN does not.

Table. 2: Multi-class Logistic Regression Performance Metrics

Accuracy	0.38
Balanced Accuracy	0.31
Precision	0.37
Recall	0.39
F1 Score	0.37
True Positive Rate	0.70
False Positive Rate	1.52

C. Convolutional Neural Network (CNN) - AlexNet

A training sequence was run on the modified AlexNet for 20 epochs with a batch size of 100 and a learning rate of 0.0001. The training loss decreases with the number of epochs and ultimately converges to 0 or a value very close to it as the number of epochs increases. After the training phase, the trained model was evaluated on the training and testing sets to obtain losses of 0.1005 and 4.1615 on the respective sets. Finally, a more comprehensive evaluation of the model was performed after running the model on the testing data by computing the various performance metrics as done before. The graph showing the training phase loss and the table with the computed performance metrics are shown below.

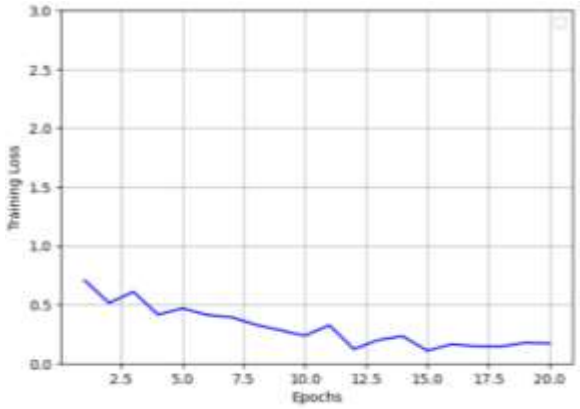

Fig 4. AlexNet Model Training Loss vs Epochs

Table. 3: AlexNet Performance Metrics

Accuracy	0.394
Balanced Accuracy	0.335
Precision	0.335
Recall	0.335
F1 Score	0.335
True Positive Rate	0.69
False Positive Rate	0.61

In terms of overall accuracy, AlexNet performed better than each of the KNNs and Logistic Regression at 39.4%. On the other hand, its balanced accuracy was similar to the KNNs at 33.5%, whereas its precision, recall, F1 score, and true positive rate were lower but comparable to both of the previous algorithms. However, the false positive rate was the metric that performed significantly better for AlexNet, with a score of 61% compared to both KNN and Logistic Regression which performed significantly worse.

The possible reasons for these results could be that AlexNet may have better learned the complex features and patterns present in the imbalanced OCT image dataset. This could have resulted in better classification performance on the majority class, which then could have contributed to better overall accuracy and false positive rate. On the other hand, and once again due to imbalance in the dataset, AlexNet may not have performed well on classifying the minority classes, which led to poorer scores in the other metrics. With AlexNet being a more complex model with higher degrees of freedom compared to KNN and Logistic Regression, its performance can be improved through techniques such as transfer learning, data augmentation techniques such as rotation, scaling, and flipping, and hyperparameter tuning such as modifying the learning rate, batch size, and regularization strength.

D. Random Forest

The Random Forest model in the sklearn library was imported and utilized for running the classification problem. A batch size of 64 was used and the n_estimators parameter was selected as 100. The training phase was run on 80% of the dataset and the testing on the remaining 20%. The model was then evaluated by computing the various performance metrics shown in the table below.

Table. 4: Random Forest Performance Metrics

Accuracy	0.79
Balanced Accuracy	0.76
Precision	0.81
Recall	0.76
F1 Score	0.78
True Positive Rate	0.87
False Positive Rate	0.26

Of all the algorithms run, Random Forest performed the best in all metrics. It had a reasonably high accuracy and balanced accuracy of 79% and 76% compared to the other classification algorithms. It also had a high true positive rate of 87% and a low false positive rate of 26%, indicating that the model can accurately identify positive instances while minimizing the number of false positives.

Although unexpected, some of the possible explanations for the better performance of Random Forest in this classification problem are that it can perform well even on an unbalanced dataset since it constructs each decision tree on a random subset of the training data, thereby reducing the impact of the majority class. Furthermore, Random Forest can capture non-linear relationships between features and the target variable, and the ensemble learning approach can combine these decision trees to make a final prediction.

IV. CONCLUSION

This project explored four different machine learning algorithms to perform Diabetic Retinopathy Severity Scale (DRSS) classification on an unbalanced OCT image dataset. The algorithms were run using Python, and a comprehensive performance evaluation of each of the algorithms was conducted. The algorithm that performed the best out of the four was Random Forest, primarily because of its strength in dealing with unbalanced datasets. However, this does not single out Random Forest as the best classification algorithm for this problem. There exists a lot of room to investigate different algorithms and CNNs with different architectures and thus this problem is still open to further exploration.

V. REFERENCES

[1] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks", in Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS), pp. 1097-1105, 2012.

[3] Tin Kam Ho. "Random Decision Forests". Proceedings of 3rd International Conference on Document Analysis and Recognition. QC, 14-16 August 1995, pp. 278-282 doi: 10.1109/ICDAR.1995.5989