Performance and Memory when Diagnosing Retinal Diseases using Classification

Rose Rodrigues Hissa Amorim - rodr0875@umn.edu

University of Minnesota - Twin Cities || CSCI 4521 - Applied Machine Learning for Computer and Data Scientists

Abstract

Classifications of images are a very important problem in the machine learning field, and a very common sub-problem is analysing labeled medical images and grouping them based on their diagnoses. There is a multitude of ways to do that, and each of them have their own advantages and disadvantages. Namely, the more advanced and flexible of a model you desire to have, generally the longer it takes to train and run them, in addition to the memory and computational effort. As such, an analysis was made of three separate models - a KNN PCA-based model, a custom-made DCNN, and a pre-made model, MobileNetV3 - and their complexity, accuracy, training time and predicting time. Each obtained accuracies of 72.6%, 86.4%, and 89.4%, while each model took 1 second, 2 hours and 20 minutes to train. Pre-made models are thusly more appropriate for the quick training and evaluation seeked.

Introduction

The maladies of cataracts, diabetic retinopathy and glaucoma are ailments that affect close to 190 million people worldwide. Their diagnoses can prove vital to a vast array of people, but there are moments when doctors might not notice specific details, and it's trivial to imagine that it's in a patient's best interest to obtain a proper diagnosis.

As such, an alert that comes up upon the scan is obtained to help the medical professional be more alert for that disease could be a great addition, but to help with adoption, it needs to be fast, efficient, and compatible with a large array of devices - possibly ones with low specifications. Thus, a question arises as to which technique would be better suited for that task.

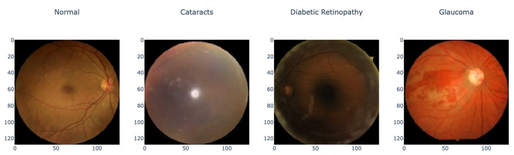

Figure X - Randomly chosen images of a normal retina followed by ones with cataracts, diabetic retinopathy and glaucoma respectively.

Methodology

Firstly, the input images were separated into a train-test-validation set with a 0.7 - 0.15 - 0.15 distribution. The randomization, to preserve results across different runs, was determined with a seed. Then, three models were chosen, and their relative accuracies were compared via a confusion matrix based on a validation set. The first model was a KNN classifier. A PCA will be done to reduce the dimensionality in the model, and the dimensions that yield a variance close to 98% will be chosen. Afterwards, the model will be tested for every K to find the one with the highest test accuracy. The second one was a custom-designed Deep Convolutional Neural Network. Its specific structure will be determined by tuning. Finally, a pre-structured model called MobileNetV3-Large - chosen for its balance between efficiency and accuracy - will be trained in the normalized dataset. For the latter two, their training process was designed to keep track of the lowest test losses, and if there are 7 instances of increased loss, early stopping will happen.

Results

For the KNN, the first 128 dimensions were chosen via the elbow method. Then, K=1 was chosen, again using the same method.

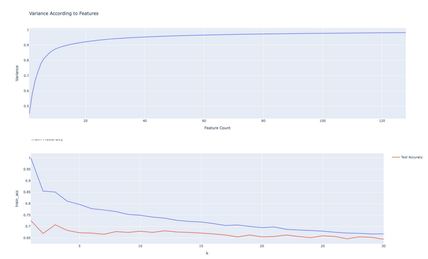

Figure X - Graph of chosen PCA-d dimensions (x) by preserved variance (y) (upper graph). Graph of training (blue) and testing (red) accuracies (y) according to K (x) (lower graph)

After multiple iterations of trial and error with the structure of the DCNN, its structure was decided to have 1 input layer, 5 convolution layers, 2 linear layers and 1 output layer.

Results

Alongside the convolutions' ReLu+Maxpools, a residual was used in the 3rd layer, and a locally-connected convolution in the 5th. It had a total of 30,450,794 trainable parameters. It was trained on at most 50 epochs with a batch size of 35.

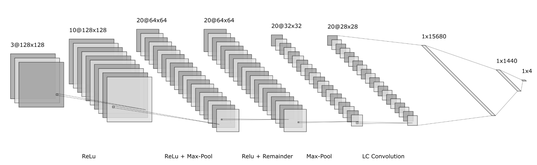

Figure X - Representation of the custom-made DCNN.

At the end, the overall accuracy in the test dataset for the **KNN**, **DCNN**, and **MNV3L** were, respectively, **72.6%**, **86.4%**, and **89.4%**. Training for each model took around **1 second**, **2 hours** and **20 minutes**. These were the obtained confusion matrices:

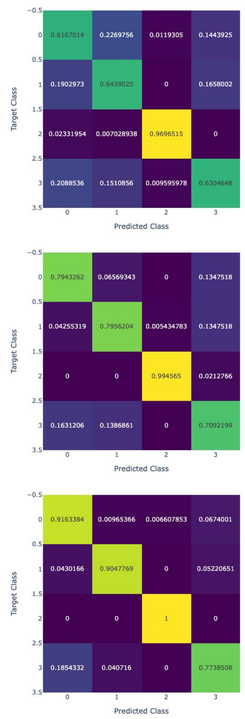

Figure X - Confusion matrices for KNN (top), DCNN (middle) and MNV3 (bottom)

It's interesting to remark that the images for diabetic retinopathy were always consistently classified with extremely high accuracy.

On a separate note, it was impressive to see that a simple 1-NN was able to achieve 72.6% accuracy in less than a second of training.

The custom-made DCNN did fare remarkably well, but its training time was the longest. It also has an impressive number of parameters, especially when compared with MNV3L's 4,207,156.

Conclusion

All three models were to correctly classify retinas more often than not, especially considering the lowest random-guessing accuracy would be 25%.

Across the entire spectrum of possible models, the most certain diagnosis is diabetic retinopathy. One of the possible reasons could be a peculiarity when it comes to its images, either because of how it affects the retina, or because of the manner in which its scans specifically where chosen.

Despite the velocity of KNN, it does misclassify a notable number of times. The custom-made DCNN was close to being the best, but its training time and almost 8x more parameters makes it so that it would not be the priority choice when adjusting for memory, performance and accuracy.

In the end, MNV3L was able to achieve the highest accuracy with still a great training speed. This tracks, since it was made with memory restrictions and lower-capacity phone CPUs in mind.

As such, if a classifier to help facilitate the diagnoses of these diseases without interfering with the amount of time a medical professional would spend is desired, it's a wiser choice to utilize verified and pre-made neural networks then making up your own from scratch - unless one has the required expertise by themselves.

Acknowledgements

The database eye_diseases_classification by Guna Venkat Doddi was obtained from Kaggle.

190 million figure from 10.1016/j.ophtha.2021.04.027, 10.1038/s41433-025-03743-z and 10.1136/bjo.2005.081224

Graphs made using Plotly.

DCNN diagram made using NN-SVG by Alex LeNail.

MobileNetV3-Large developed by Howard et al (10.48550/arxiv.1905.02244), imported via TorchVision.