Milestone 3: Implementation

Objective:

The primary goal behind this ocular disease detection project is to train a convolutional neural network to classify fundus images, digital photos from inside the human eye, into several distinct diseases. Over the course of multiple milestones this goal has grown in detail and so has the list of requirements. These details consist of both data-specific information as well as overall architecture decisions for training. This paper will discuss all these issues, their implications on each other, and how they ultimately affect performance throughout the model's development.

Deliverables:

The deliverable at hand was a trained classification CNN with an accuracy above 85% over its last few stages before being submitted to GitHub (the largest public repository for sharing code). As such, it is important that there is a clear understanding about where the success of this model stems from along with any potential areas for improvement. To aid in answering these questions I present my own personal experiences through writing down what went right or wrong during certain periods of time within this project while also discussing some common practices which have worked best for me when trying to improve upon models' performance. In addition, this document includes links towards more detailed discussions regarding aspects of data collection and processing which are not discussed here but nevertheless still highly relevant given their effect on overall results. Furthermore, I provide general tips toward improving model architecture design base. The chosen web-based user interface application is Streamlit, which is a platform known for its data analytical visualizations, and this application will now be expanded upon in terms of its utilized features.

Mapping of Functional Requirements

The following table outlines the functional requirements required to meet the end goal of this project. It should be noted that although these individual components may seem unrelated, they were designed by taking into consideration future functionality improvements needed after each milestone is completed.

* Training: Trains the CNNs using various techniques based on loss curves observed.
* Testing: Takes the final pre-trained model and performs evaluations via holdout tests on test sets.
* Deployment: Imports the trained model to run locally, then deploys it onto Google Cloud for hosting.
* Data Collection: Uses ODIR to gather image data for training purposes.
* Data Processing: Converts raw images from different sources into JPG format; processes image dimensions and labels them appropriately.
* Model Architecture: Chooses specific layers used within convnets which would allow better generalization between classes.
* Hyperparameters: Sets hyper parameters according to previous experience, current state of available hardware, as well as desired level of complexity for future versions of the project.

Source Code Listing

GitHub Link: <https://github.com/DouglasBui/GCU/tree/main/Capstone_Project>

Code Review

Graphical user interface, text

Description automatically generatedGraphical user interface

Description automatically generated

Graphical user interface, website

Description automatically generated

Implementation Plan

The plan for implementation is beset through the Streamlit python application framework. Here we can see that every major step is broken up into steps involving either data preparation, training, or testing. After each stage is complete, one moves on to the next until finally arriving at the deployed model ready to be tested against live patient data in the Streamlit platform. Unfortunately, there has been a series of bugs and limitations that has prevented the CNN model from being transferred over to the platform, hence why the final section remains blank. Although this does pose a problem, the solution seems quite simple, just copy paste the file contents directly from the source computer into another environment running the same version of Python. Once the data is uploaded to the webpage, the user should be given a options to move to patient analysis tab to examine each individual patient's ocular digital images and diagnosis. The next tab is equipped to handle mass amounts of data that will return a csv file back to the user with a completed list of diagnosis. The final tab is testing and validation, where the user can verify the accuracy of their results by performing tests and viewed sets of comparisons.

Instructions for Streamlit Webapp

was by far the most laborious and intensive part of the project. It took a few hours

Project Requirements Review

The first phase of requirements in developing this project involved collecting data. This part consisted mostly out of gathering images of patients whose conditions had been diagnosed. For this process I relied heavily on two programs called ODIR (Open DIabetic Retinopathy) and ODRC (Open Diabetic Retinopathy Consortium). They allowed easy access to thousands of high quality retina photos taken across many different hospitals around the world. The next set of requirements was a proper vetting and standardization of all source images that would be fed into the model. This was by far the most laborious and intensive part of the project. It took a few hours alone to draft up a system that removed unnecessary black space from each image. This was important for increasing the information value within each segmentation. Another set of requirements dealt with training the model itself, which entailed selecting the number of layers as well as choosing the learning rate and batch size. The latter proved especially challenging due to the fact that small variations caused drastically larger changes in the output predictions. A good rule of thumb is that if you increase your batch size by 10 times, you can expect the error rate to go down by 0.5 times. Finally the testing requirements still have a lot left to desire, due to the lacking of complete functionality with my implementation, it is rather difficult to show the user multiple models at this time. Hopeful this can be solved soon within the next week.