Fernando Zambrano
Professor Jafari
Machine Learning II
December 9, 2019

Final Project: Individual Report

Individual Work:

As part of a group of two, I shared half of the project responsibility. The focus of my work was on network background research, data processing, and coding. Given the focus on biology in our exams, I chose to continue this theme with our final project centering on Diabetic Retinopathy classification.

Data Collection:

My first task was to download the dataset. There were a few issues that needed to be resolved. As my partner and I's first competition we didn't have any experience collecting Kaggle data. It was my responsibility to discover a way to get the entire dataset comprised of over 82 gigabytes split among 15 datasets and get the Kaggle.json token on our GCPs. Since the data competition was relatively old, the Kaggle API did not allow for all of the data to be download at once on the GCP. Additionally, there was no description about the number and classes of images that were in each file. Even if my partner and I downloaded one file, our local machines would not have enough space to hold it before uploading to cloud. To avoid possibly missing any crucial data I had figure out a Linux command that would allow us to get the data on GCP. The command used is below:

```
for f in $(kaggle competitions files diabetic-retinopathy-detection | cut -d ' ' -f 1 | tail -n+3); do kaggle competitions download diabetic-retinopathy-detection -f $f; done
```

Research:

Before any coding was started, I researched popular pretrained networks and techniques that other competitors had applied on the dataset. The most successful networks applied was the *Oxfordnet,* but it was not a pretrained model available in Keras. Instead I decided to use the Resnet50 model as it was the winner of the 2015 ImageNet competition, the same year which the Kaggle competition took place. The goal was to compare our results with the competition leaderboard, but as we would later find out there were issues with our model which would make a comparison inaccurate.

Some of the key findings that I found out we should implement was the image size to feed our network. The image sizes were large are about 1000 x 1000, however with such size and large dataset, our network would run slow. Instead, the ideal image used was 256 x 256. In addition to dataset specific research, I dedicated time to researching deep learning

Data Processing:

Since the bulk of our data was running on my machine, I was in charge of data processing and running the models on my computer. My coding process was to first create a basic Resnet50 model and then fine tune the parameters afterwards. This proved to be much more difficult as I began to run out of time because of the amount of time spent on discovering a preprocessing method to import the images and their associated labels. The issue with preprocessing was that there was only one lable.csv for all of the images, but not all 35,126 images could be used as a consequence of memory constraints. I was able to develop a few functions to subset the data to 7,920 images so that the resulting NumPy arrays would fit onto memory and classes would resembled the original data distribution.

After getting the data into NumPy array format, I fed the NumPy arrays into ImageGenerator to provide data augmentations such as flips, shears, and rotations. The remainder of the code my partner and I exchanged implementing different optimizers, learning rates, and activation functions code to render the Resnet50 architecture more successful. My partner played a larger role in developing the code to plot our model history an evaluation.

Results:

Our model consistently produced a training accuracy score of 74.3%. In terms of validation accuracy, our model performed well as it would peak at 82.2% but would have a training and validation loss of 0 for 30 epochs. Furthermore, our other evaluation scores consistently produced a Cohen Kappa of 0.0, and an f-1 score about 0.18.

Conclusion:

Overall, our model produced a decent accuracy score, but given the feedback from our Cohen Kappa and f-1 scores, our model still needs development. We concluded that since I sub-set our training data to mirror the original data distribution, I missed the opportunity to balance the data by increasing sample size of underrepresented classes such as 1,3, and 4 which had less than a couple hundred samples in the sub set. If given the opportunity to redo this project, I would look into better methods of loading and preprocessing the training data such as a data loader to avoid memory constraints on the GPU and also dedicate more time focusing on hyper-parameters and network architecture.

References:

Chollet, Francios, (2018). Deep Learning with Python. Shelter Island, NY: Manning Publications Co.

Diabetic Retinopathy Detection (Dataset). (20150. Retrieved from: https://www.kaggle.com/c/diabetic-retinopathy-detection

Dwivedi, P. (2019, January 04). Understanding and Coding a ResNet in Keras. Retrieved from: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33