The Parkinson Pathfinder

Aditya & Mishra
1009807327, adits.mishra@mail.utoronto.ca
Link to Project.

Project Description

Parkinson's is a complex disease that challenges the lives of many in the way that it works. A big part of detecting Parkinson's and detecting it early is to look at MRI scans of individuals and identify abnormalities that indicate Parkinson's. The rationale behind this project is to create a model which can utilize the brain scans of individuals and determine whether they have Parkinson's disease or are healthy. Image classification models that can understand subtle patterns that radiologists might miss.

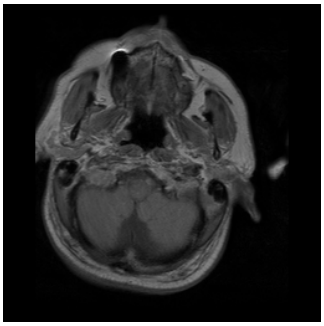

Figure 1: Example of a Parkinson Brain Scan

As can be seen, MRI brain scans can be incredibly complex, and so a model can help a radiologist more easily determine whether someone has Parkinson's or not. This report will provide an overview of how the project has progressed. Specifically, what data processing has been done, what the baseline model is, and what the approach to the primary model is going to be. There are also going to be a series of diagrams to show the performance of the baseline model, and primary model.

Data Processing

I got my data set from Kaggle where I found 831 images of brain scans. Both of patients with Parkinson, and those without it. Thankfully due to the nature of the data set, I didn't have to do much cleaning. I had to emphasize data augmentation, resizing and normalization as the two main mechanics for my portion of data processing.

The first step of my data augmentation was rotating images. One of the problems with my data set is that I only have 831 pieces of total data. I realized that by slightly tilting images, I could double the size of the data set. So I performed a random angle tilt on all the images and increased my dataset size. However, this was done in a limited capacity between -30 and 30 degrees. Obviously patient brain scans wouldn't be completely flipped or rotated 90 degrees. Hence I decided on 30 degrees.

The next step was to ensure my data was a reasonable pixel value, and to normalize it. This would be done so that my model can be trained efficiently and capture appropriate features. For this I chose an input size of 128x128 pixels as mentioned in my proposal. The reason I normalized is because I want the scanner to focus on features, not intensity, and this will enable the learning of disease patterns. A key factor to mention at this point is that I had to normalize differently for the baseline model then we have historically done. Finally, for my split I'm using an 80-20 split. The test is the 20%. From there I'm also splitting the 80% into training and validation data sets. With me keeping about 20% once again for validation.

Baseline Model and Architecture

For the architecture of my baseline model, I'm using a SVM. The SVM is a model that is more often then not chosen for healthcare imaging and heavily so for classification tasks. The way the SVM works is by looking for the best hyperplane that divides the two classes between Parkinson's and Normal. Historically speaking SVMs provide robust accuracy in medical imaging classification tasks.

In my case, I used 10 K-fold iterations and a random state of 24 to ensure repeatability. The next portion shows graphics which illustrate how the baseline model performed. I have a graph, and Confusion Matrix.

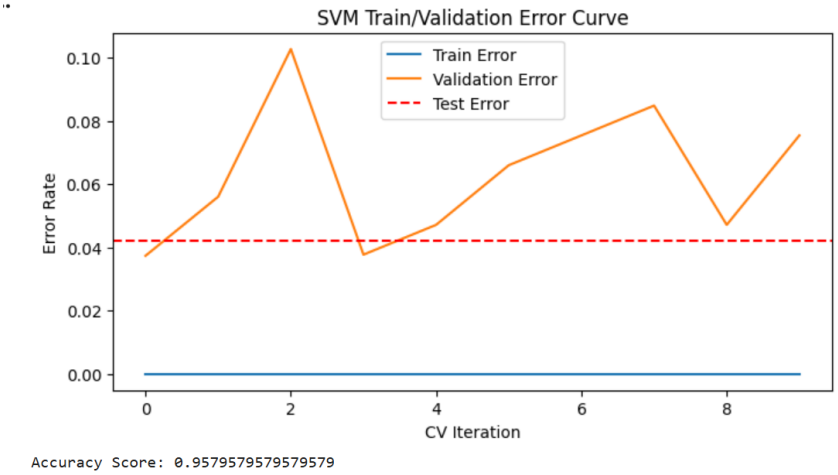

Figure 2: Graph of Errors with Respect to the Iteration

It can be visually seen that the SVM model performs very well on the test data with the error being minimal and not changing. We can also see that I printed out the accuracy just below and that ends up being nearly 96%. This allows us to conclude that the SVM model works quite well when we look at our data set. We can also see that this is the case in the confusion Matrix we which created through the code. It is displayed below.

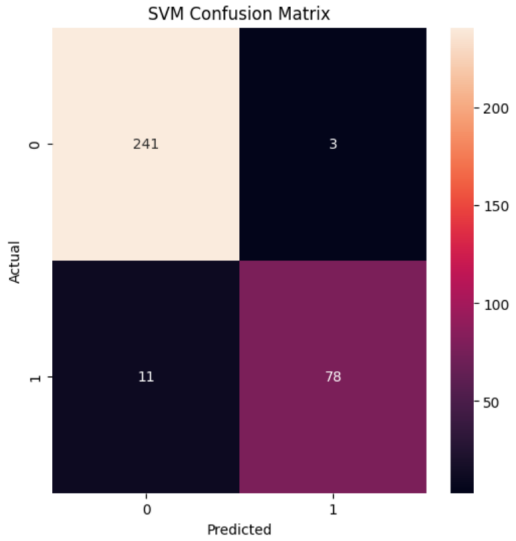

Figure 3: Confusion Matrix of SVM

The matrix shows how accurate the whole model is, and shows the baseline works very well. We can see that the accuracy is very high, and this is good, but looking at the dataset we also know its

limited by quantity, and thus might not be the most reliable. The loss is also incredibly low, and this shows that the model is working, but could use improvements. The final portion I wanted to speak to was a difficulty I encountered. I realized in my proposal I said I was going to use ANNs, but that was because I was most familiar with them at the time. When I did some more research, and through my projects works, I realized SVMs would be better. This meant changing the work from the previous stage. Additionally, I wasn't super familiar with SVMs, and as a result this made it much more challenging to implement.

Primary Model Work

The primary model I constructed was a "Test" CNN. I used an architecture of three convolutional layers, two pooling layers, and three fully connected layers. For the end instead of the initial sigmoid I proposed I decided to use a multi-classification system. As talked about in my proposal the reason I decided to go with a CNN was because they can address class imbalances, and are highly accurate. Especially at image recognition. I also was able to create a graph of testing over 20 Epochs. I will speak more to the quality of the tests momentarily. The learning rate used here is 0.001 and the batch size is 16. The main metrics of evaluation which were plotted were the training error, and the training loss.

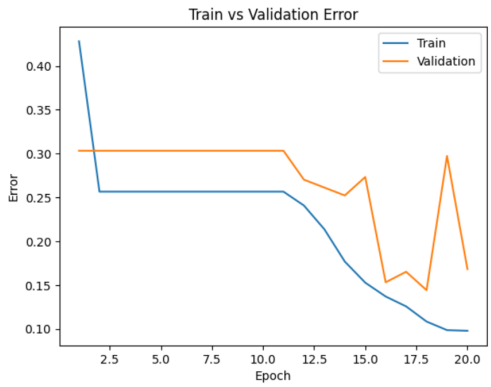

Figure 4: Train vs Validation Error of the CNN

Figure 5: Train vs Validation Loss of the CNN

As can be seen through the curves there's clear instances of the model fluctuating, but largely we can see that the loss is decreasing over time which means the model is training. I can also see that there may be a case of overfitting due to the specific parameters I've used. We can see this could be possible because of the sudden fluctuations happening across the Epochs. My initial intuition tells me I need to increase the batch sizes, and also potentially the learning rates.

In my opinion this was a successful test model, but clearly appears to be a case of overfitting. I think a challenge with this is going to be tuning the hyper parameters. I found that the time taken was exceptionally long due to the large number of features, and I clearly need to refine it further for the model to be considered accurate. This was something I planned, but I believe the emphasis should be on the baseline and the data processing.

Conclusion and Next Steps

The next major steps are definitely for me to tune the hyper parameters. I think the SVM showed me a strong case, but I want to make sure I can tune my CNN appropriately to be able to identify the data. The model definitely is still low fidelity, but I believe the case has merit, and I'll be able to successfully complete the project.