Using Machine Learning to Detect Alzheimer's Disease in MRI Scans

Sam Lizotte

Table of Contents

1. Abstract

Alzheimer's Disease (AD) is a neurological disorder that slowly eats at the brain and affects a patient's memory, thoughts, and behavior. It often develops in the later stages of life, and is a very heartbreaking disease to watch a loved one go through. As neurologists get closer to finding a cure for Alzheimer's Disease (AD), it's still necessary to catch the disease in its early stages to ensure the best quality of life for those who have it. Along with quality of life, it's important to know if a patient has AD in order to protect them from autoimmune disorders that can worsen their symptoms or pose a threat due to their vulnerable state. We aimed to answer the question about if Magnetic Resonance Imaging (MRI) scans, which are often used in the diagnosing of other neurological disorders, can be used to diagnose AD in patients. While neurologists have already attempted this, we wondered if we could take it a step further by using machine learning to classify the data and separate it into different categories of dementia in order to properly diagnose a patient, as well

as determine the severity of their AD. While conducting this research, we found surprising results, shown in how one of our inferior models reported nearly perfect accuracy while a model that was supposed to be superior reported a significantly poorer accuracy. Overall, our models reported back accuracies over 70%. These models also had to be incredibly complicated, with the image data being put into it having to be processed through hundreds of layers if we wanted a satisfactory accuracy. We concluded that while machine learning did report back very high accuracy, it is not a perfect tool for detecting fallacies in MRI scans, it is not meant to be the sole diagnostic for AD.

2. Introduction

Alzheimer's Disease (AD) is a neurological disorder that affects around 500,000 people per year. This disorder is a form of dementia, damaging the brain and affecting a patient's memory, thinking, and behavior. The disorder gets severe enough to the point where it interferes with the patient's daily life, eventually shutting down their brain. It is currently incurable; however, diagnosing the disorder early can help with preventing it and ensuring the patient a higher quality of life. There are many methods neurologists use to detect this disorder, including brain scans such as PET scans and MRI scans. MRI scans are typically not seen as a tool that would be solely used to diagnose AD, but our research aims to see if machine learning can change this fact. By taking our dataset, which uses vision data, and plugging it into multiple categorical models, we aimed to see the accuracy in which a machine could detect the difference between brains with Alzheimer's and brains without Alzheimer's. We even included different stages of dementedness, ranging from none to moderate severity. This problem is a supervised classification model, where the images are processed through and sorted into different categories. Hopefully, with this data, we can work on Alzheimer's detection in order to keep those with Alzheimer's as healthy as possible.

3. Background

Magnetic Resonance Imaging (MRI) scans are not enough to diagnose Alzheimer's Disease (AD) alone. Researchers are working on this problem, making different types of MRI models that will one day be the sole diagnostic tool used to detect Alzheimer's. One example of this is the qGRT MRI scan, although it is still in its early stages. The qGRT MRI scan is a subset of MRI scans that can detect signs of dementia in patients in under 10 minutes. These qGRT MRI scans are focusing on patients with early stages of AD rather than late stages, due to the necessity in

diagnosing the disease early. Currently, the qGRT MRI scan is not enough to diagnose on its own; however, researchers are working on developing it so that one day it can replace other diagnostic methods such as spinal taps or PET scans. PET scans are similar to MRI scans, although they are much more costly and the results take longer to receive. Spinal taps can cause pain due to the removal of cerebrospinal fluid from the lower back, and because of this are considered more "invasive" than a brain scan is. The qGRT MRI scan presents a solution that is both less expensive, quicker, and easier on the patient. These models look for fallacies in the brain in the form of dark matter, which would signify deterioration in the brain caused by dementia. These models may still be in the works, but are showing exponential improvement and may one day be solely used to diagnose AD, showing that it is possible for MRI scans to detect early onset AD in a patient.

4. Dataset

The dataset used in our project was a 6,400 image dataset, and all images were used. The data was vision-based data, due to the fact that it consisted of images. These images were Magnetic Resonance Imaging (MRI) scans (although they were not qGRT MRI scans) from patients with different severities of dementia, which were separated into four classes: no signs of Alzheimer's, very mild Alzheimer's, mild Alzheimer's, and moderate Alzheimer's. There were no MRI scans with severe Alzheimer's Disease. There were also only 64 images with moderate severity, which could serve as a potential bias because most of the images were either slight cases of Alzheimer's, or no signs of Alzheimer's. In fact, most of the image data was of those with no signs of Alzheimer's; however, there was still a 50/50 split between the number of images without signs of dementia and the number of images with any signs of dementia, no matter the severity. In the non-demented class, there were 3,200 images. There were 2,240 images in the very mild demented class, 896 images in the mild demented class, and again, 64 images in the moderate demented class.

This data was collected from several public repositories, including hospitals, and put together into one set. All images are in 128x128 format, and have already been pre-processed before being put into the dataset. Because the images were already pre-processed, the only thing left to do before classifying the images was making the numpy arrays so that the computer knew which images had what classification.

The normalized images acted as the X_array in the training data, while the classification y_arrays made with the assigned numbers were the reference arrays. The data was split into 75% for training, or 4800 images, leaving 25%/1600 images for testing data. The data was also shuffled to ensure that both training and testing data had images from every label possible, to prevent incorrect classification which would lead to a faulty model with poor accuracy.

5. Methodology/Models

Four machine learning models were used in order to classify the data: decision trees, logistic regression, neural networks, and convolutional neural networks.

Decision Trees

Decision trees are classification models with different layers that the images pass through. Similarly to a tree map, which looks at all possible outcomes of a question, decision trees start with an initial problem. Based on the answer, or the details found in the image, the decision tree makes different "decisions" that classify the images into different categories. Decision trees are simpler models, as adding too many layers can make them too complex and result in overfitting. Overfitting is when a model gets too used to its training data, and in turn, performs poorly when the testing data is run through it. This defeats the entire purpose of the model, so it is important to avoid overfitting in models. In the case of the decision tree, it's to make sure that the data is processed through as few "nodes," which process and categorize the data, as possible.

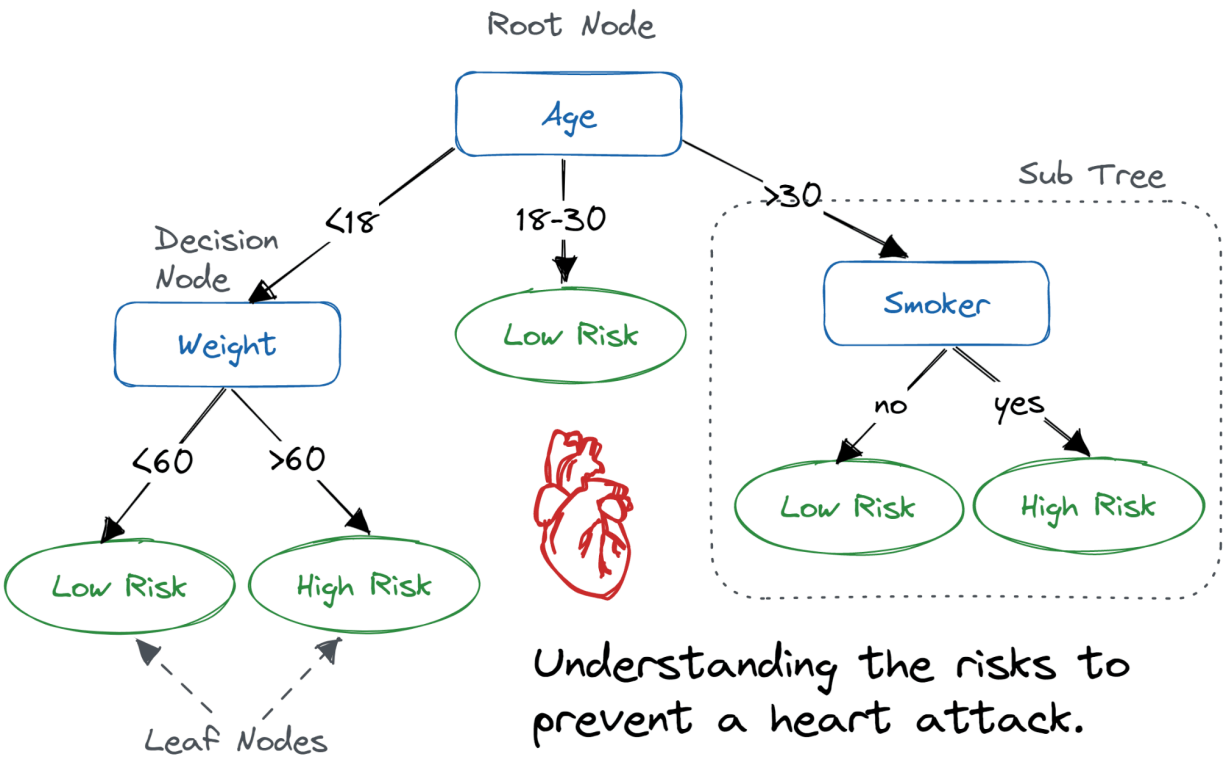


Diagram of a Decision Tree

The root node is the question, or dataset put into the decision tree model. The internal nodes act as the decision makers, while the leaf nodes are the categories that the image data is separated into. For the decision tree to make these choices on which images go where, a calculation for entropy is used. Entropy is a value between 0 and 1 that determines how an image should be classified before being passed through to the next set of nodes, and decision trees aim to find the smallest entropy possible in order to get the highest accuracy. Decision trees want the smallest entropy possible because this means there are the least amount of impurities in the data and it is being correctly, and uniformly classified.

Logistic Regression

Logistic regression is a type of machine learning model that classifies data by measuring the likeliness of it being in a certain class. Logistic regression is a more complex form of linear regression, although it is typically not seen as a very advanced model, either. Logistic regression uses an equation of "log odds" to determine the odds, or likelihood, of a piece of data being in a certain class. The "log odds" equation is: Logit(pi) = 1/(1+ exp(-pi)), and ln(pi/(1-pi)) = Beta_0 + Beta_1*X_1 + … + B_k*K_k. Logit(pi) is the output, and x is the input. This log odds equation is run

through the dataset using a for loop for a specified amount of iterations. For this project, we used a logistic regression model with 200 maximum iterations, and then one with 1000 maximum iterations. This model specifically used nominal logistic regression, due to the fact that there were four possible outcomes for the data to be put into. If the categories had any specific required order to them, then the data would need an ordinal logistic regression model, but since all categories hold equal value, the data used the nominal logistic regression model.

Neural Networks

Neural networks are a type of classification model that use different layers of "neurons" to classify and separate data into different categories. Neural networks are modeled after the human brain, which is why the nodes that images pass through are labeled neurons. Each node has a weight attached to it, which is usually randomized during training while the computer is learning which types of images go where. The dataset is put through the input layer, and then passed through a certain number of hidden layers before being classified in the output layer. The number of hidden layers depends on the model, and is typically considered a hyperparameter due to how often it is manipulated to test accuracy. Our model used 200 hidden layers.

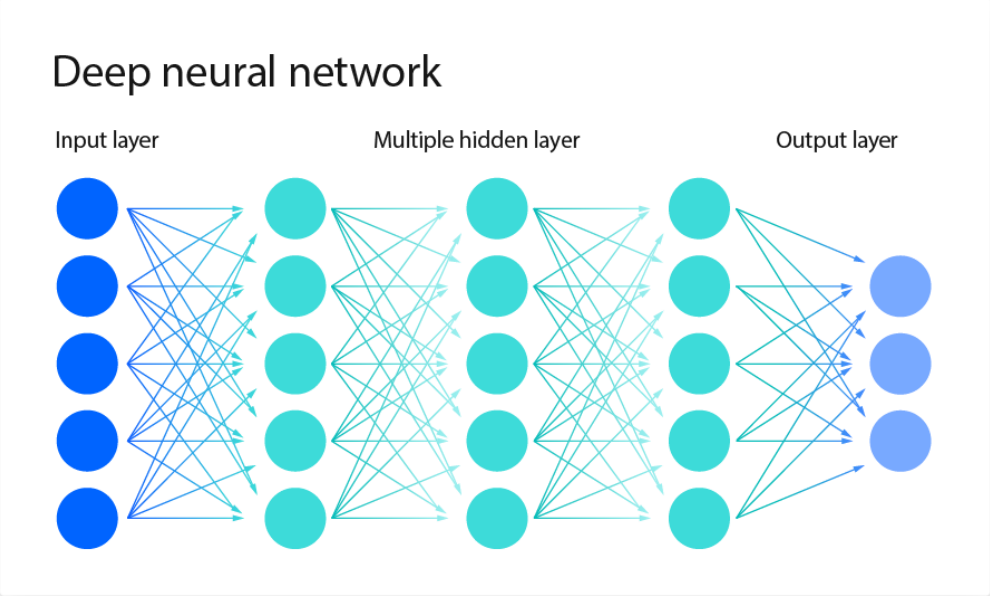
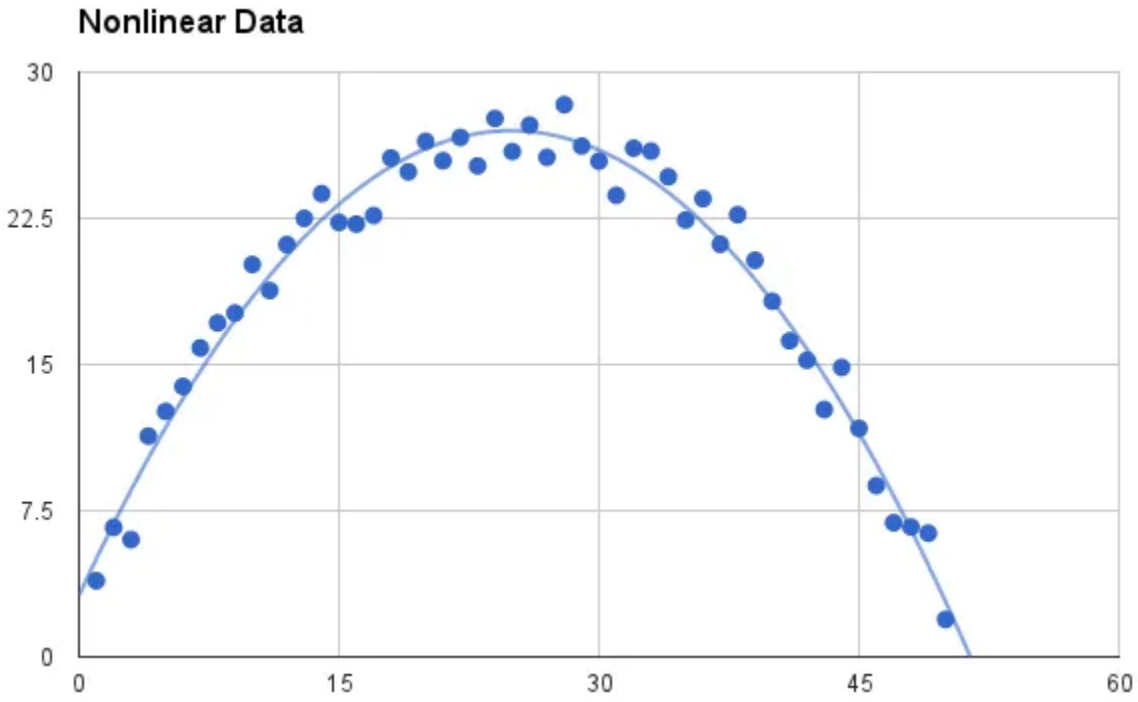


Diagram of a Neural Network

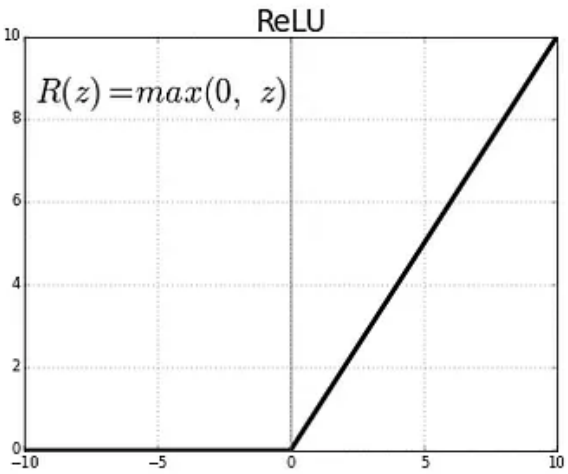
The formula for a neural network looks like this: Σwixi + bias = w1x1 + w2x2 + w3x3 + bias and output = f(x) = 1 if Σw1x1 + b>= 0; 0 if Σw1x1 + b < 0. Each neuron looks at a certain matrix of pixels to look for similarities or differences when classifying. The bias refers to the weights of the nodes, and the 1x1, 2x2, etc. refer to the pixel matrix that the neurons are "looking" at with computer vision.

In order to properly function as a neural network, non-linear activation functions are required to determine the output of the model. Non-linear activation functions, as stated in the name, add a non-linear component to the neural network which makes the model more complex. The equation used to do so is $y = \alpha(w1\ x1 + w2\ x2 + \ldots + wn\ xn + b)$, with α as the activation function that is triggered to create this non-linearity. Without these functions, the input data would simply be passed through one layer, which would be no different than putting input data through a linear regression model.



Graph of a Non-Linear Activation Function

Visually, these functions make the output data appear as a curve rather than a straight line. This makes the data more flexible, which therefore increases the quality of the model as a whole. In our neural network, we used the Rectified Linear Unit (ReLU) function and the softmax function.



Graph of a Rectified Linear Unit function

The Rectified Linear Unit function is a rather simple function in its composition, but it significantly improves the performance of the neural network model when classifying data. Since the ReLU function not only produces linear outputs, but can also return true zero values, it is easier for the computer to read and therefore the most optimal activation function to use in a neural network model. But these true zeroes can also pose a problem, as the ReLU function cannot process negative input values and creates the issue of "dead" neurons in the model. If a neural network has "dead" neurons, then it will not process certain data inputted into the model, and therefore will return an incorrect output; however, variations of the ReLU function have been created to quell this problem.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a more complex form of neural networks that have the same outline as a neural network, with an input, output, and hidden layers that images pass through. However, CNNs have other layers besides those of neurons that help make it superior when it comes to specifically classifying data. Convolutional layers, which have neurons just like the hidden layers in a neural network do, convolute the images in order to make them easier for the computer to read. By simplifying the images, the computer can more easily find the differences in images that would put them into different classifications. CNN models typically have two convolutional layers that are activated by Rectified Linear Units (ReLU), which help make the convolution process easier. Our model used two convolutional layers, with different amounts of neurons in them acting as the hyperparameters.

Convolutional layers also have pooling in them, which reduce the amount of things that the computer is looking at by grouping, or "pooling" certain things together. Our CNN model used MaxPooling, which sends the pixel with the highest

value to the output layer. The pixel with the highest value is determined by the matrices of pixels that the neurons look at, being combined into one value with the convolutional layer.

Our model also used a BatchNormalization layer, which normalized the data and therefore made it easier for the data to pass through the computer. Although our data was already normalized, BatchNormalization layers, which go after the convolution layers, normalize the images even more. Our model then used a dense layer with a value of 128, which runs the data through the equation **output = activation(dot(input, kernel) + bias)** to classify the data. Before passing our images through the dense layers, we flattened the data in order to run it through the dense layers more smoothly.

While training the data, the model used a certain amount of epochs to run the data through before being sent to testing. Epochs are similar to the amount of trials that a CNN runs, sorting through the data and reporting an individual loss and accuracy for each epoch, as well as a validation loss and a validation accuracy. Loss is a unit of measurement to test how well the model is evaluating the data. It is important to make sure that loss is the smallest value possible, while accuracy is the highest value possible.

VGG19 Convolutional Neural Network

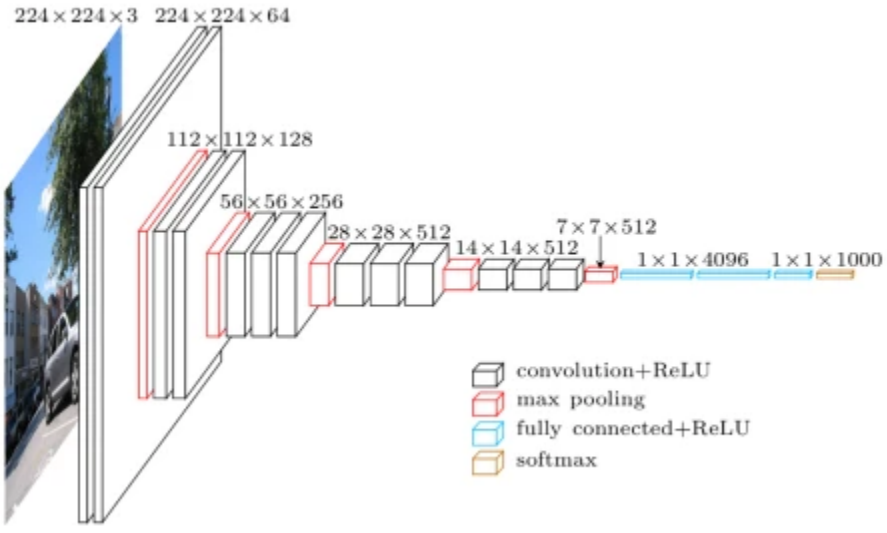


Diagram of a VGG Convolutional Neural Network

Compared to a regular convolutional neural network model, which only uses 2-4 convolutional layers, the VGG19 model (VGG stands for “Visual Geometry Group”)

uses 19 convolutional layers. This classifies it as a “deep” model, meaning that it uses incredibly small kernels to fully look through and classify image data. Due to their complexity and intricacy, these models are better for larger datasets but can also be incredibly time-consuming due to the size of the model. These models also require more epochs to properly train the data, adding to the amount of time it takes to fully train the data before it can be tested.

6. Results and Discussion

Results with Decision Tree Model

The first model used, the decision tree, reported an accuracy of ~70% the first time, and then 68% the second time. This was expected as the decision tree model was not that complex, and no hyperparameters were modified to see if the accuracy would get any better.

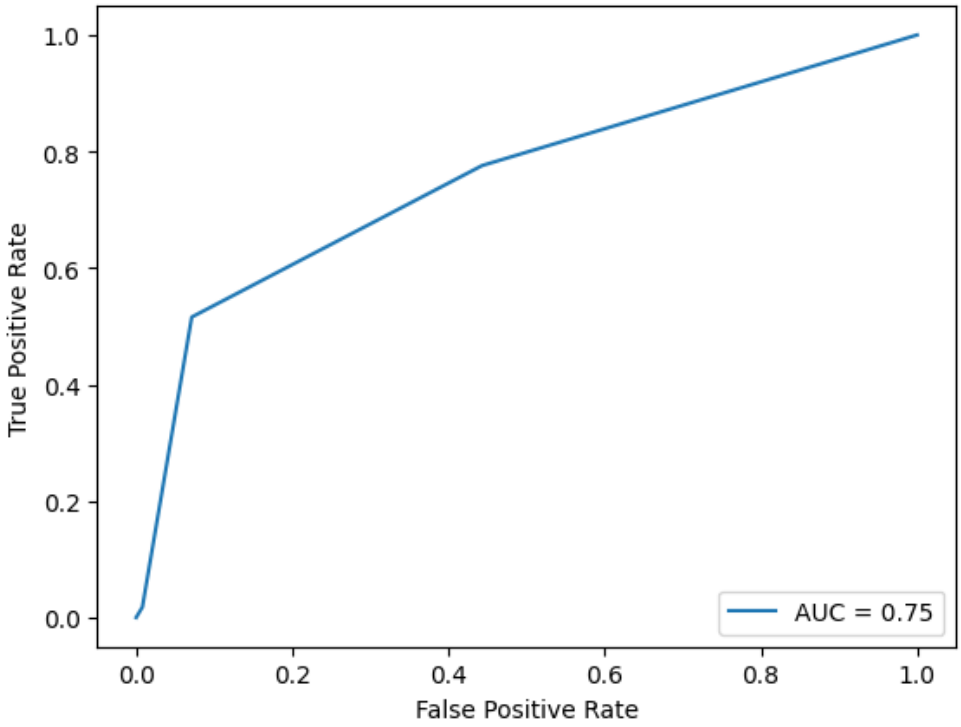


Figure 1 - Decision Tree ROC Curve

The receiver operating characteristic (ROC) curve for the decision tree is a way of measuring true positives and false positives. The numerical values on the x and y-axis refer to the accuracy, showing which images were correctly classified as true positives compared to which images were incorrectly classified as false positives.

Our ROC curve showed the area under the ROC Curve (AUC) was 0.75. While this value is not bad, it's not the best AUC that could have been given by the model. The closer that the AUC value is to 1, the better the model, because this would indicate that all values have been classified correctly and were true positives in the model.

Significant Accuracy in Logistic Regression

The second model, the logistic regression model, reported back with the highest accuracy, increasing from an accuracy of around 50% the first run to 97% the second run. The difference between these two models was the maximum amount of iterations used. Due to the fact that the dataset was so big, the model could not run through all of the data with merely 200 iterations, resulting in a poor accuracy. The iterations in this model was the hyperparameter that we experimented with, and although 1,000 iterations still did not completely run through the data, it was enough to report back a near perfect accuracy. This accuracy seemed a bit too good to be true, and we suspected that something went wrong in the model to report back such a good accuracy.

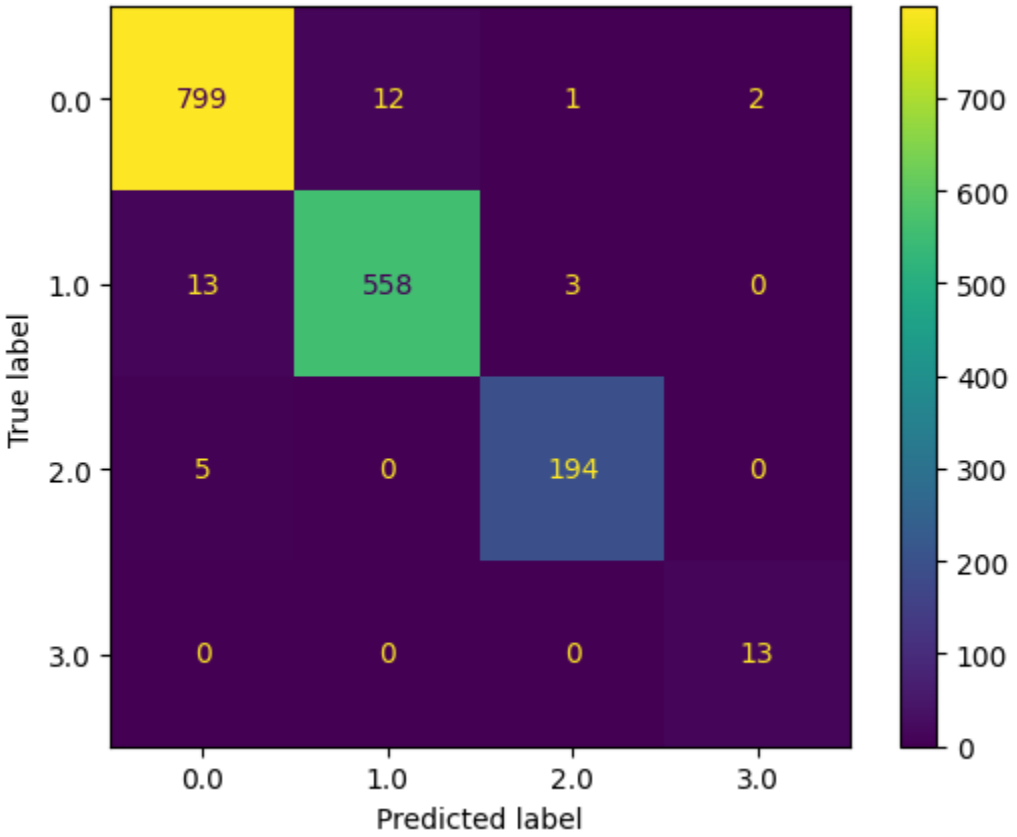


Figure 2 - Logistic Regression Confusion Matrix

We used a confusion matrix to look at the results of the logistic regression model. The confusion matrix is another chart used to detect false and true positives, as well as false and true negatives. The numerical values in the x and y-axis refer to the values that we assigned the different categories of images from our dataset when we processed and put the data into our y-array. Therefore, this confusion matrix shows that 799 of the non-demented MRI scans were correctly classified, 558 of the very mild demented images were correctly classified, 194 of the mild demented, and 13 of the moderate demented. 13 may not seem like a large value, but recall on the fact that we only had 64 images of moderately demented patients, and most of those images had been put into the training data. This confusion matrix is reporting on the accuracy for the testing data, meaning that it is going off of fewer images run through the computer. The confusion matrix has many 0s in it, meaning that most images were correctly classified, which reflects the 97% accuracy reported earlier from the test set data. This confusion matrix debunks the idea of anything going wrong in the data, showing that most images were actually classified correctly. While this is good news for our data, it also begs the question as to why our logistic regression did so well as a model, while our CNN still did very poorly.

Results in Neural Network Model

As expected, the neural network model did impressively well, reporting back with an accuracy of 87% the second time, and then 91% the third time. The hyperparameter in this model was the amount of hidden layers. Like the logistic regression model, when only 30 hidden layers were used, the neural network model reported back with an accuracy of around 33%. We quickly identified the problem being that the model didn't have enough layers to properly process the data, and changed the number of 30 hidden layers to around 200 hidden layers. While this was a big jump, our model reported back with an accuracy of 87%, only getting better after running our data through the model once more.

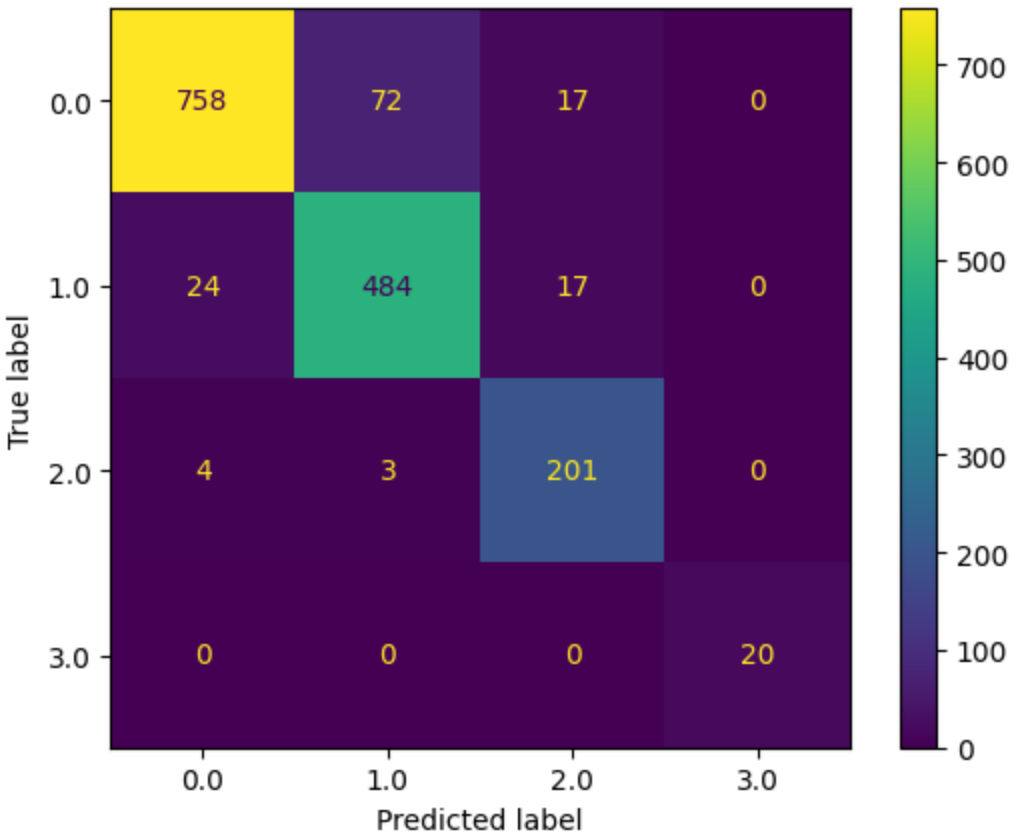


Figure 3 - Neural Network Confusion Matrix

Like in the logistic regression confusion matrix, most of the images were classified correctly in the neural network model. The numerical values on the x-axis and the y-axis are the same as they were in the y-array and in the confusion matrix. This confusion matrix is telling us that in the testing data, 758 images were correctly classified as non-demented, 484 as very mild demented, 201 as mild demented, and 20 as moderate demented. Compared to the logistic regression model, the neural network model was better at determining which MRI scans had more severe cases of dementia compared to those that had less severe cases or none at all. Although the difference is small, it is still important to note. The neural network also had more misclassifications compared to the logistic regression, but this is shown in the accuracy as the neural network model has slightly worse accuracy compared to the logistic regression model.

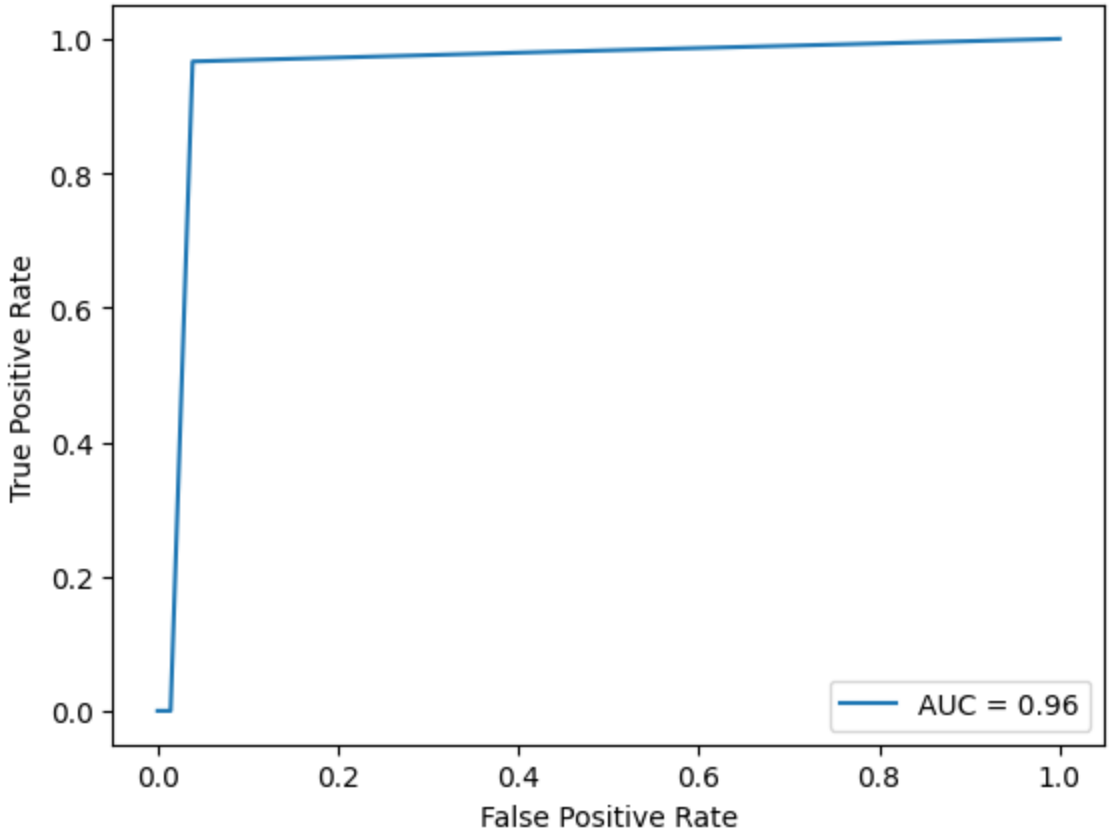


Figure 4 - Neural Network ROC Curve

We also used a ROC curve for our neural network model, which turned out much better than the ROC curve used for the decision tree. The AUC was 0.96 compared to 0.75, which means that more data was correctly classified as a true positive rather than a false positive. While this is not perfect, that is good, because if the accuracy were perfect, then that would mean that there was a problem with the data- potentially overfitting. It makes sense that the AUC for the neural network is much better than the AUC for the decision tree, due to the fact that the neural network reported back a significantly better accuracy and was a better model to use for this data.

Shocking Results in Convolutional Neural Network Model

The convolutional neural network (CNN) reported the worst accuracy by far, which was incredibly surprising because the CNN was expected to be the best model. Originally, the CNN repeatedly reported back accuracies in the 40s, but eventually was brought up to accuracy in the 50s. While this accuracy could reflect on a faulty model, it could also reflect on the fact that the machine is simply not fit to

process Alzheimer's in MRI scans and must be worked on before it is trustworthy. However, due to the amount of models that reported accuracies over and around 70%, we are more likely to believe the former.

The CNN itself had 12 layers, which may not have been complicated enough for the caliber of the data being fed into it. Similarly to the logistic regression model, the CNN may have needed more layers to properly process the amount of data that was being put into it. Our model used Rectified Linear Units (ReLU) to activate the convolutional layers, and had two convolutional layers processing the data. We first tried 32 neurons in the first convolutional layer, and 64 in the second. After that, we tried 64 neurons in the first layer, and 96 in the second. We decided on keeping 32 and 64 as the values of neurons in the convolutional layers, due to the fact that the second model with more neurons was reporting an even poorer accuracy. This could be a sign of overfitting, which meant that the model probably needed less convolution because the images were becoming too simple for the computer to classify properly. Each convolutional layer had a BatchNormalization layer after it, meaning the data was normalized before being sent to the second BatchNormalization layer, and then finally put through the dropout and dense layers. Once again, because the data was flattened, normalized, and convoluted so much, it could have been too simple for the computer to properly read.

While training, the model had between 10-13 epochs, as the number of epochs also served as a hyperparameter. This could have been too few epochs for the computer to properly read the data and classify it correctly before getting sent to the testing process. The validation data was the testing data used in all of the other models.

```
Epoch 1/13
38/38 [==============================] - 21s 179ms/step - loss: 1.9474 - accuracy: 0.3348 - val_loss: 2.2983 - val_accuracy: 0.4944
Epoch 2/13
38/38 [==============================] - 6s 150ms/step - loss: 1.4034 - accuracy: 0.4817 - val_loss: 2.2897 - val_accuracy: 0.4944
Epoch 3/13
38/38 [==============================] - 6s 145ms/step - loss: 1.2739 - accuracy: 0.4873 - val_loss: 2.2804 - val_accuracy: 0.4944
Epoch 4/13
38/38 [==============================] - 6s 146ms/step - loss: 1.2136 - accuracy: 0.4779 - val_loss: 2.2686 - val_accuracy: 0.4944
Epoch 5/13
38/38 [==============================] - 6s 145ms/step - loss: 1.1944 - accuracy: 0.4819 - val_loss: 2.2554 - val_accuracy: 0.2556
Epoch 6/13
38/38 [==============================] - 5s 145ms/step - loss: 1.1703 - accuracy: 0.4825 - val_loss: 2.2400 - val_accuracy: 0.2056
Epoch 7/13
38/38 [==============================] - 5s 143ms/step - loss: 1.1523 - accuracy: 0.4906 - val_loss: 2.2214 - val_accuracy: 0.1637
Epoch 8/13
38/38 [==============================] - 6s 146ms/step - loss: 1.1511 - accuracy: 0.4879 - val_loss: 2.1995 - val_accuracy: 0.2837
Epoch 9/13
38/38 [==============================] - 6s 148ms/step - loss: 1.1373 - accuracy: 0.4956 - val_loss: 2.1718 - val_accuracy: 0.3537
Epoch 10/13
38/38 [==============================] - 5s 141ms/step - loss: 1.1287 - accuracy: 0.5025 - val_loss: 2.1365 - val_accuracy: 0.3562
Epoch 11/13
38/38 [==============================] - 5s 143ms/step - loss: 1.1206 - accuracy: 0.5102 - val_loss: 2.0897 - val_accuracy: 0.3531
Epoch 12/13
38/38 [==============================] - 5s 141ms/step - loss: 1.1071 - accuracy: 0.5042 - val_loss: 2.0289 - val_accuracy: 0.3550
Epoch 13/13
38/38 [==============================] - 6s 147ms/step - loss: 1.1028 - accuracy: 0.5094 - val_loss: 1.9463 - val_accuracy: 0.3581
<keras.src.callbacks.History at 0x7ee440be2e00>
```

Figure 5 - CNN Epochs

For this training, we used a split of 75/25 training and testing data for the CNN to run through. Surprisingly, this reported a worse accuracy than when the data used a 50/50 split and a 70/30 split. This could potentially be a sign of overfitting, due to the oversimplified model and the large amount of training data it was fed.

```
score = cnn.evaluate(X_test, y_test)

print("Loss:", score[0])
print("Accuracy:", score[1])
```

```
50/50 [==============================] - 1s 12ms/step - loss: 1.9463 - accuracy: 0.3581
Loss: 1.9462952613830566
Accuracy: 0.3581250011920929
```

Figure 6 - CNN Evaluation

This evaluation, from the same run as Figure 5, shows the validation accuracy from Figure 5 as well as the validation loss, rather than the regular loss and regular accuracy. This could be a sign of overfitting because the training accuracy was around 20% higher than the validation accuracy, which used the testing data.

Improving the Convolutional Neural Network

Due to the fact that the convolutional neural network was supposed to be our best model, we were not satisfied with the fact that it was reporting only 50% accuracy. In order to fix this, we rather easily traced the root of the problem to the simple fact that the model was not complicated for the amount of data that we were feeding it. The reason that the model performed poorly when more neurons were added to its existing layers was because there were simply not enough layers, nor enough epochs to properly run the data in order to return a satisfactory accuracy. So, we added two more layers, and increased the amount of neurons in each layer. At first, we attempted three convolutional layers: the first had 64 neurons, the second 128, and the third 256. Each chunk of convolutional layers had its own batch normalization layer and max pooling layer. The amount of epochs was also increased to 100 so that the model would be properly trained and handled without leaving anything out. When the model reported back with a 58% accuracy, we knew that we were on the right track of improving the CNN in order to allow it to properly work as it should.

After this test run, we added another convolutional, batch normalization, and max pooling layer. The CNN now had four convolutional layers: 64 neurons in the first, 128 in the second, 256 in the third, and 512 in the fourth. The amount of epochs was increased to 150, and this model reported back with 72% accuracy– a significant increase. From there, we kept the amount of neurons in the convolutional layers the same, but once again increased the epochs by 50. With 200 epochs, the model reported back with 77% accuracy. While this accuracy was extremely better than what we originally began with, it was still not enough.

We then retraced the lack of accuracy to the fewer number of neurons in the convolutional layers, and went back to the model to double each value. Now our CNN once again had four convolutional layers, but the amount of neurons went as follows: 128, 256, 512, 1028. After running this model, the accuracy given back was 91%– an astounding change from the 77% reported when the amount of neurons in each layer was half of what they were now. From here, we wondered if we could get the model to report a higher accuracy, so we once again increased the amount of epochs by 50. With 250 epochs, the accuracy increased only slightly, now being 93%.

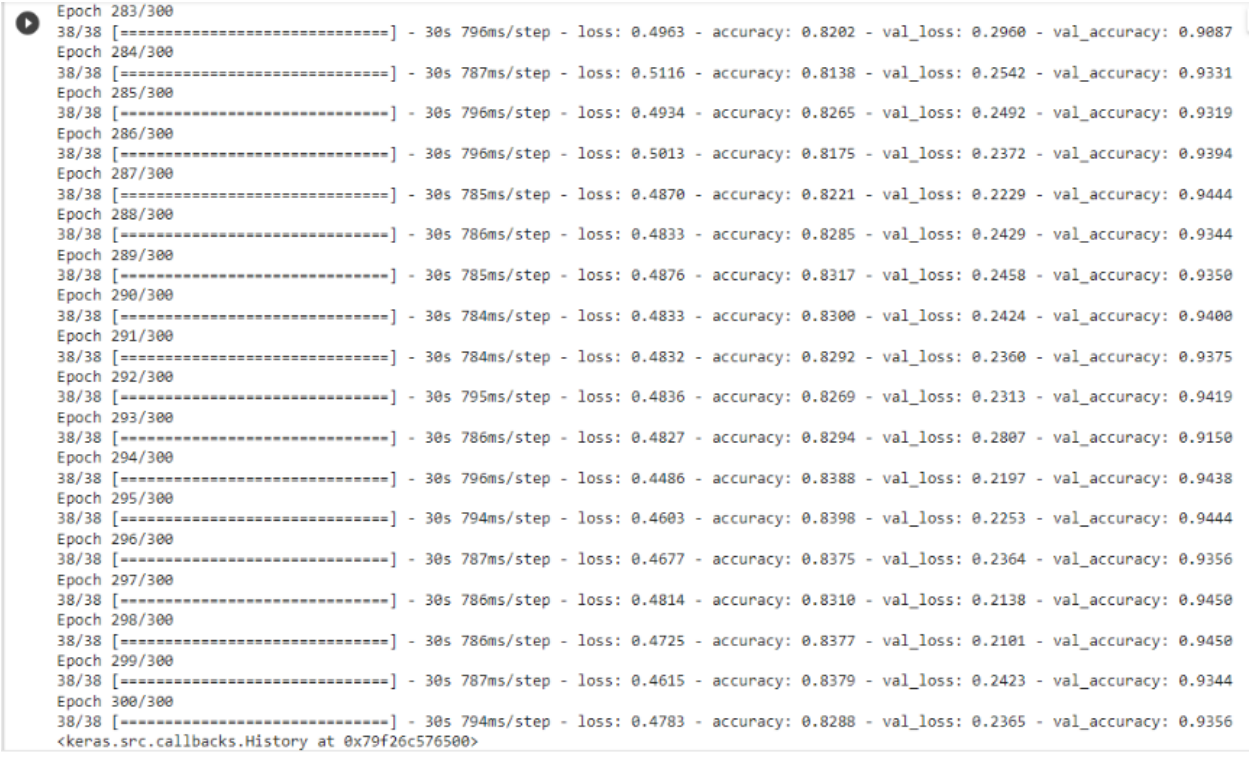


Figure 7 - CNN Improvement (300 epochs)

Figure 7 shows us once again wondering if we could get any more accuracy out of the model, so it was run with 300 epochs. The change in accuracy was less

than 1%, which meant that we had reached a true accuracy of 93%. This accuracy was much better than the ~50% that the model had been reporting when it was first run, but understandable once we considered the size of the first models compared to the size of the dataset that it was attempting to analyze and train with.

Surprising Accuracy in VGG19 Convolutional Neural Network

After running our regular convolutional neural network model, we decided to put our data into a VGG19 CNN model in order to see if the accuracy would increase or stay the same. We did this by keeping the model that we already had, with the only difference being that the manually added convolutional, batch normalization and max pooling layers were deleted and replaced with the VGG19 model which added 19 convolutional layers. We also decreased the amount of epochs by half, because we were unsure of what the model would do. But decreasing the amount of epochs quickly proved to be a mistake due to the sudden loss of accuracy, as the model reverted back to 50% accuracy. This was a disappointment, as the VGG19 accuracy was supposed to thoroughly look through the data and report back with a higher accuracy. After analyzing the model, and taking into account the complexity of it, we hypothesized that the problem lied in the fact that there were not enough epochs to properly process the data and report a correctly represented accuracy. Due to lack of resources, we were not able to test this hypothesis, but we came up with another possible hypothesis which was the idea of overfitting. Due to the fact that the model performed incredibly well with only 4 convolutional layers, the idea that the VGG19 model had gotten too used to the training data from the dataset provided is a very feasible possibility and that is why it reported back with such poor accuracy compared to its predecessor model.

7. Conclusion

A potential problem with this research could be presented in the fact that there is not enough moderate Alzheimer's Disease being represented in the dataset. There is no severe Alzheimer's Disease at all, which would weaken the model if it were presented with an MRI scan displaying severe dementia in a patient. However, this project aims to search for mild Alzheimer's in order to detect it early, which means that it is a positive that a large portion of the data lies in very mild or mild severity. The non-demented MRI scans act as a control group, and the large quantity ensures that hopefully there are less false positives or false negatives that would lead to incorrect diagnoses. There is also a possibility that the overwhelming amount

of non-demented MRI scans leads to a bias, which means that the data is more likely to present a false negative or false positive.

We hope to have a GPU so we can properly run our VGG19 model with more epochs in the future, because we are interested in whether this was the problem that prevented it from performing as high as the other convolutional neural network models that only had four convolutional layers. We hypothesize that 500 epochs would significantly increase the accuracy, although it still may not be enough to properly represent the data and we still may not get an accuracy that is satisfactory. But all we truly need is for more epochs to show better accuracy, because this will prove our hypothesis correct and then we can build off of those results.

The lack of a GPU made it so that we were not able to run the models without the platform they were on crashing due to overload of RAM. In order to get a GPU, we needed a subscription, which cost money that we unfortunately did not have. If we were to get this money, and this GPU, then the models would have been able to perform to their full potential, and could have been evaluated more fairly.

Overall, our logistic regression model performed the best at 97%, while our VGG19 convolutional neural network (CNN) model performed the worst at ~50%. Incidentally, these were our most shocking results as well, due to how unexpected the inverse correlation of model complexity and accuracy was when comparing these two models.

The CNN model initially performing poorly was also shocking to see, because they did just as poorly as the VGG19 models when they were first being run. This parallel strengthens the hypothesis that the complexity added to the model causes the increase in accuracy, because once the regular CNN had been improved and more epochs had been added, it quickly became our second best model with an accuracy of 93%.

In the future, when we better our model, we want to see if we can use it to find a correlation between the existence of Alzheimer's Disease (AD) in a patient and the severity of an autoimmune disease the patient gets. We would most likely compare a patient's MRI scans to COVID data, looking at the patient's lungs after COVID and see how damaged they have become. Although this project would require many more variables, because we would need not only classification based on severity of Alzheimer's, but classification based on severity of COVID as well. Currently, there is no publicly available dataset that has both COVID information and

AD information in the same patient, meaning that the potential to work with this type of data cannot now be reached.

8. Acknowledgements

We would like to thank the Inspirit AI mentorship program for providing resources for making this research possible. We would also like to specifically thank Ivan Villa-Renteria for providing mentorship and advice so that this research could be properly carried out and conducted in the form of machine learning.

9. References

Alzheimer's Facts & Statistics. Alzheimer's San Diego. (2022, September 21). https://www.alzsd.org/resources/facts-stats/

Everding, G. (2022, March 2). *Damage early in alzheimer's disease id'd via novel MRI approach*. Washington University School of Medicine in St. Louis. https://medicine.wustl.edu/news/damage-early-in-alzheimers-disease-idd-via-novel-mri-approach/

Ji, X., Lian, Y., Dong, G., Ding, X., & Liu, X. (2023, August 15). *Comparison of Brain MRI Findings between Patients with Alzheimer's Disease and Non-Dementia Psychiatric Disorders in the Elderly*. Research Square. https://www.researchsquare.com/article/rs-3259508/v1

Kumar, S., & Shastri, S. (2021). Alzheimer MRI Preprocessed Dataset. Retrieved August 9, 2023,.

Plater, R. (2022, June 25). *How an MRI Brain Scan May Help Diagnose Alzheimer's Disease*. Healthline. https://www.healthline.com/health-news/how-an-mri-brain-scan-may-help-diagnose-alzheimers-disease

Boesch, G. (2023, March 16). *VGG very deep convolutional networks (vggnet) - what you need to know*. viso.ai. https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/

Hardesty, L. (2017, April 14). *Explained: Neural networks*. MIT News | Massachusetts Institute of Technology. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

Having a lumbar puncture. Alzheimer's Society. (n.d.). https://www.alzheimers.org.uk/research/take-part-research/lumbar-puncture

Keras Team. (n.d.). *Keras documentation: Dense layer*. Keras. https://keras.io/api/layers/core_layers/dense/

Navlani, A. (2023, February 23). *Python decision tree classification tutorial: Scikit-Learn Decisiontreeclassifier*. DataCamp.

https://www.datacamp.com/tutorial/decision-tree-classification-python

The Pennsylvania State University. (n.d.). *12.1 - Logistic Regression*. 12.1 - Logistic Regression | STAT 462. https://online.stat.psu.edu/stat462/node/207/

Simonyan, K., & Zisserman, A. (2015, April 10). *Very deep convolutional networks for large-scale image recognition*. arXiv.org. https://arxiv.org/abs/1409.1556v6

What are Convolutional Neural Networks?. IBM. (n.d.-a). https://www.ibm.com/topics/convolutional-neural-networks

What are Neural Networks?. IBM. (n.d.-b). https://www.ibm.com/topics/neural-networks

What is a Decision Tree?. IBM. (n.d.-c). https://www.ibm.com/topics/decision-trees

What is Alzheimer's Disease? Alzheimer's Disease and Dementia. (n.d.). https://www.alz.org/alzheimers-dementia/what-is-alzheimers

What is Logistic Regression?. IBM. (n.d.-d). https://www.ibm.com/topics/logistic-regression

What is Overfitting?. IBM. (n.d.-e). https://www.ibm.com/topics/overfitting

Yathish, V. (2022, August 4). *Loss functions and their use in neural networks*. Medium. https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9

Hebert LE, Weuve J, Scherr PA, Evans DA. Alzheimer's disease in the United States (2010-2050) estimated using the 2010 Census. Neurology. Available at www.neurology.org/content/early/2013/02/06/WNL.0b013e31828726f5.abstract. Published online before print, Feb. 6, 2013.

Hirtz D, Thurman DJ,Gwinn-Hardy K, Mohamed M, Chaudhuri AR, Zalutsky R.How common are the "common" neurologic disorders? Neurology 2007;68:326-37.

World Alzheimer Report 2015 the Global Impact Of Dementia An Analysis Of Prevalence, Incidence, Cost and Trends, p.7 http://www.alz.co.uk/research/WorldAlzheimerReport2015.pdf, and World Alzheimer Report 2015 the Global Impact Of Dementia An Analysis Of Prevalence, Incidence, Cost and Trends Summary Sheet, p.1 http://www.alz.co.uk/research/WorldAlzheimerReport2015-sheet.pdf

Alzheimer'sDisease International World Alzheimer Report 2010: The Global Economic Impact of Dementia: Executive Summary," Prof Anders Wimo, Karolinska Institutet, Stockholm, Sweden Prof Martin Prince, Institute of Psychiatry, King's College London,UK. Published by Alzheimer'sDisease International (ADI) 21 September 2010

Chng, Z. M. (2022, August 5). *Using activation functions in neural networks*. MachineLearningMastery.com. https://machinelearningmastery.com/using-activation-functions-in-neural-networks/

Sharma, S. (2022, November 20). *Activation functions in neural networks*. Medium. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Panneerselvam, L. (2023, October 19). *Activation functions and their derivatives - A Quick & Complete Guide*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/#h-nonlinear-activation-functions

Krishnamurthy, B. (2022, October 28). *An introduction to the ReLU activation function*. Built In. https://builtin.com/machine-learning/relu-activation-function