CPSC 8650
TEAM 6
Sahithi Tatineni
Sri Chandana Kovelamudi
Holly Elizabeth Amell

FINAL PROJECT REPORT

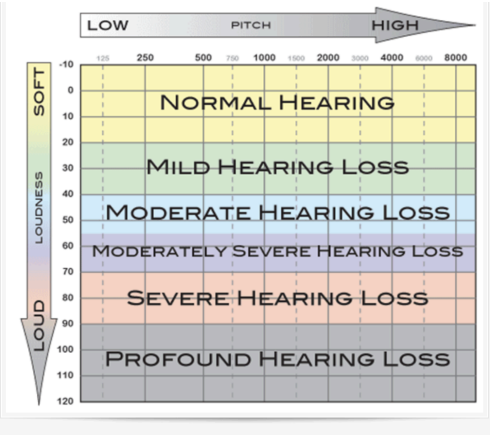
THE PROBLEM

Hearing loss is a medical issue that affects many people around the world. It can be a result of many different factors such as infection, loud noise, or various medical conditions. Metabolic and sensory components have been linked to age-related hearing loss (https://pubmed.ncbi.nlm.nih.gov/35064426/). Brain MRIs are used to take images of the brain for various medical purposes. From these MRIs, gray matter images can be created using specialized processing techniques. These gray matter images can be used to monitor the brain and the areas associated with hearing loss.
Hearing loss is measured with pure tones. It uses the quietest sound, in decibels, a person can hear. The severity of the hearing loss is measured through the lowest decibel the person can hear.

Various data mining techniques can be used to analyze the gray matter images to predict hearing thresholds. The dataset includes the gray matter image and its two corresponding pure tone frequencies, PT 500 and PT 4000. Three different data mining techniques were trained to predict the PT 500 and PT 4000 values for the gray matter images.

Background Research

According to the National Hearing Test, “hearing thresholds are the lowest level of sound that can be heard 50% of the time.” Hearing loss is represented by the difference in an individual's hearing threshold and the average for normal sensitivity. The following chart represents different hearing thresholds for different stages of hearing loss.



Hearing Loss Chart (Hz vs dB).

Our Programming

To solve this problem we used python, with the help of the libraries nibabel and keras. Nibabel is a python library that can open .nii files and extract the data from them. Keras is a deep-learning library for python and allows for intuitive creation of neural networks. Training is abstracted and the history of loss and accuracy during training is available to plot easily. Also used python's matplotlib library to plot charts and images that we accrued throughout the script.

Our Data and Data Preparation

Dr. Wang supplied the class with a dataset of gray matter images of Brain MRI scans.In total, there were 171 MRI scans which were .nii files. A csv file with pure tone values and corresponding file name with 3 fields are given:

- Field 1 (ID): name of the file the pure tone values belong to
- Field 2 (PT500): the hearing threshold at pure tone 500
- Field 3 (PT4000): the hearing threshold at pure tone 4000

Files are loaded and sorted to correspond with csv file data. Images are downsampled to decrease the size of the data then they are adjusted the data range to within specified minimum(-1000) and maximum(400) values. Scaled the data between 0 and 1 to standardize and make it consistent. Then the scan is resized to desired dimensions and the resized scan is rotated by 90 degrees. The rotated scan is reshaped to include a single channel using tensorflows 'expand_dims' function and then the images are loaded in the form of an array.

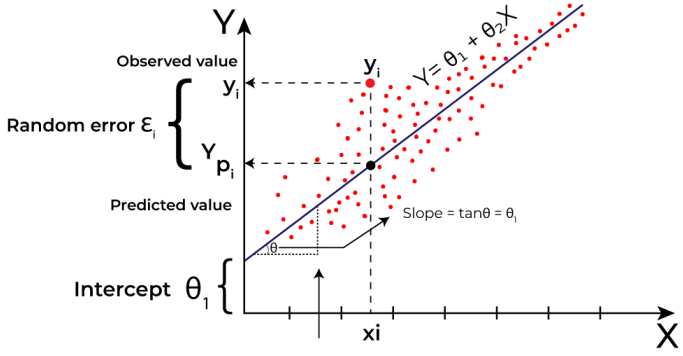
MODELS

1. Linear Regression

Linear regression serves as a valuable method for exploring the linear association between a dependent variable (Y) and one or more independent variables (X). It's crucial that the dependent variable Y is continuous, while the independent variables can be either categorical or continuous. A prudent initial step is to evaluate the potential relationship between two continuous variables through a scatter plot. Linear regression is best applied when this relationship appears to be linear, offering insights into the strength and direction of the association. However, if the relationship seems nonlinear, alternative techniques should be considered for further investigation.[1]

Linear regression provides a simple and interpretable way to examine the linear relationship between input features, such as hearing data and gray matter image features, and the target variable, which is pure tone frequency values. It serves as a foundational model for comparison with more advanced techniques like SVR and CNN, offering a baseline for assessing model performance.

Linear Regression offers clear coefficients for each input feature, aiding in understanding the linear link between the features and the target variable. Nevertheless, its reliance on a linear relationship might overlook intricate data patterns, and it struggles with non-linear relationships.

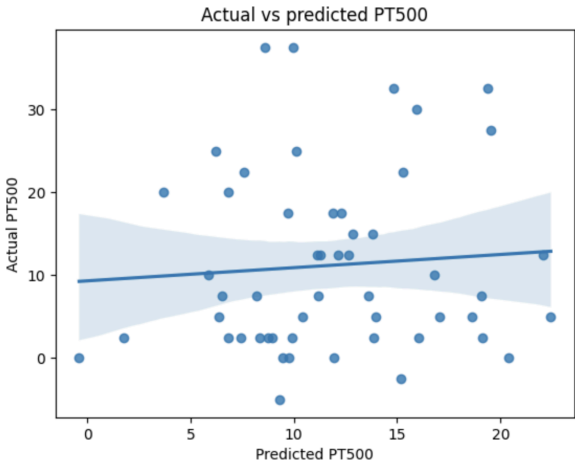


Linear Regression [2]

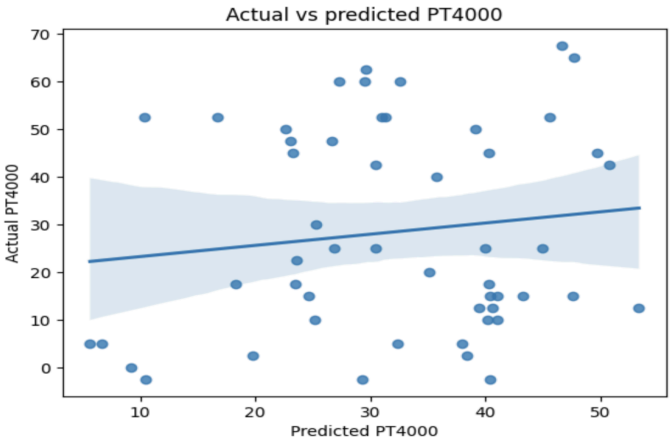
We have treated PT500 and PT4000 as two separate problems. The images were downsampled and normalized before training the model. The normalized images and PT500 values were split into two groups: train and test. The test size is 30% of the original dataset. The learning parameters ‘X_train’ and ‘y_train’ are used to train the linear regression model. X_train is the data pulled from the train images. Y_train is the corresponding PT500 values. Similarly, the normalized images and PT4000 values were split into two groups: train and test. The Test size is also 30% of the original dataset. Once again, the learning parameters ‘X_train’ and

‘y_train’ are used to train the linear regression model. The model was evaluated on the following metrics : Mean absolute error, R2 score, Mean squared error.

Results:



PT500 Actual vs Predicted Values



PT4000 Actual vs Predicted Values

Evaluation metrics of Linear regression models:

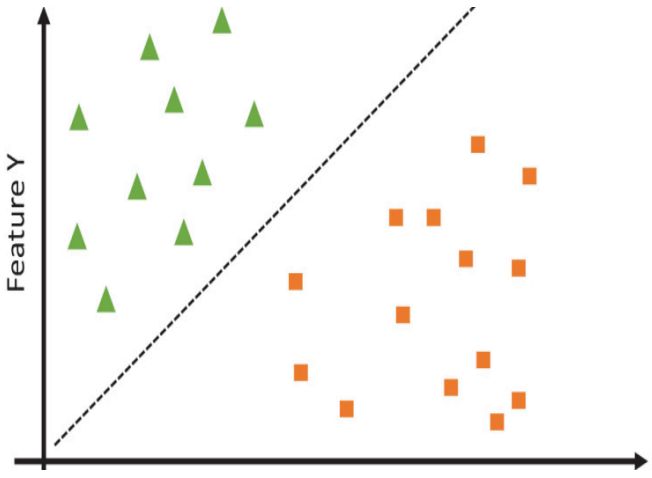
Linear Regression	Pt500	Pt4000
R2 score	-4.27	-2.89
MAE	9.29	19.78

MSE	**132.83**	**534.88**

This model performed better on PT500 than PT4000. This can be seen by the lower R2 score for the PT500 model. However, it was not particularly precise for either of the values. The mean absolute error represents how far the predicted values are from the input values. For the PT500 model, the average error was around 9.267 and for the PT4000 model the average error was 19.778. As shown in the chart from the National Hearing Test above, the difference in these values is the difference between different stages of hearing loss. Overall, this model does not precisely output the extended PT values from the gray matter images.

2. Support Vector Regression

Introduction: Data mining techniques are becoming more significant for supporting classification and forecasting tasks in the medical industry. The support vector machine is similar to the vector regression machine, which applies the same principle to regression situations. SVMs are frequently used to organize and classify the challenges. It's not always clear how SVMs are used in regression. SVRs are such models. Regression analysis is challenging because it requires determining a function approximating the mapping from the input domain to the actual number in the training sample concept. Taking into account points inside a decision's bounds is the primary goal of SVR. [3]

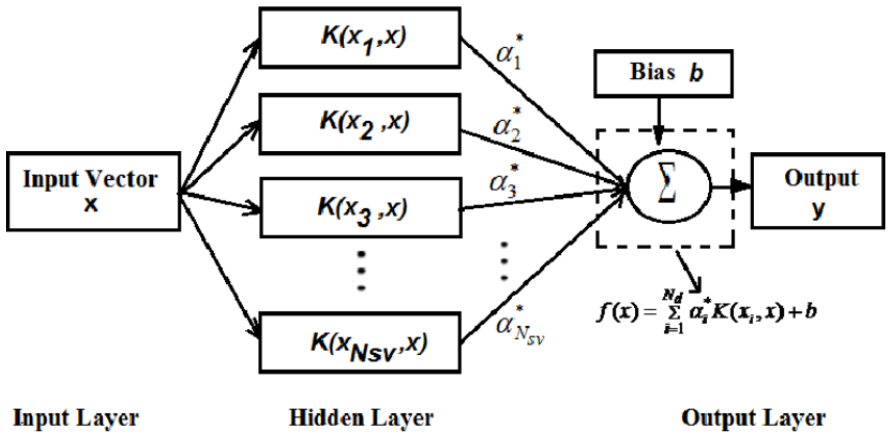


Support Vector Machine [4]

Application:

Support vector Machine(SVM), specifically SVR (Support Vector Regression), can be applied to predict pure tone frequency values. Choosing the kernel function and parameter affects the performance of the SVM model for the medical data we have as input. The kernels used were linear and RBF(Radial Basis Function). SVM is utilized to map input data into higher dimension feature space by using non linear mapping.

In SVM training, the loss of poorly classified data is normalized by the regularization parameter C. In this project, after randomly experimenting with several values of C, such as 0.1,1,10,100. C value was set to 100 randomly. SVM presents the idea of an inner product kernel function to obtain this nonlinear mapping function. In high-dimensional space, the inner product operation problem is effectively solved by the kernel function. This effectively resolves the nonlinear classification problem.



Structure of Support Vector Machine [10]

Linear Kernel:

The function for the linear kernel is given as,

$$k(X, X_i) = X.X_i$$

The inner product of the input vectors is computed by the kernel function K.[11] The relevant example's SVM output value is equal to the total of these inner product and weight combinations. In the study PT500 and PT4000 values are considered separately. Initially, for

threshold values based on the frequency Pt500, the test size was considered as 0.2. The training data was gray matter extracted from medical images and test data was threshold values of PT500. The evaluation metrics that were used were R squared score, mean squared error(MSE) and mean absolute error(MAE). The similar process was repeated on threshold values of PT4000 Hz.

RBF Kernel:

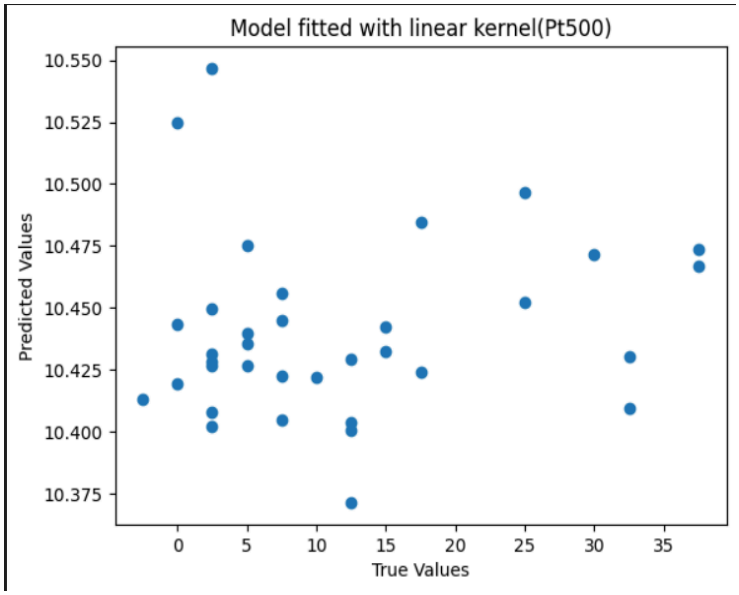
The function for RBF kernel is given as , [12]

$$k(X_i, X_j) = exp(-\gamma||X_i - X_j||^2), \gamma > 0$$

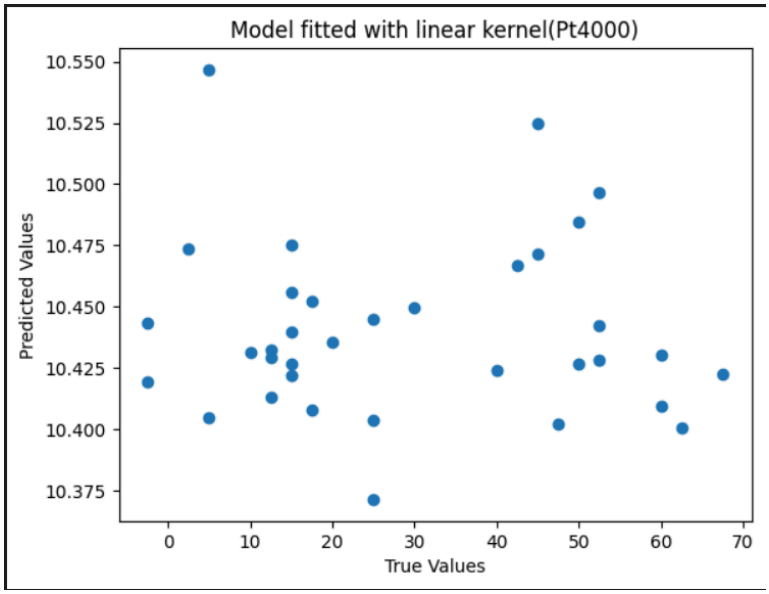
To improve the outcome, the parameters were adjusted. The value of C is 100, while gamma stands for 1/n parameters. When applied to local points, it works great. Higher C values are used to identify the training data more precisely, and gamma indicates the maximum distance that any individual parameter can travel. The training and testing data is considered similar to the SVR model with a linear kernel.

RESULTS:

MODEL WITH LINEAR KERNEL(PT500):

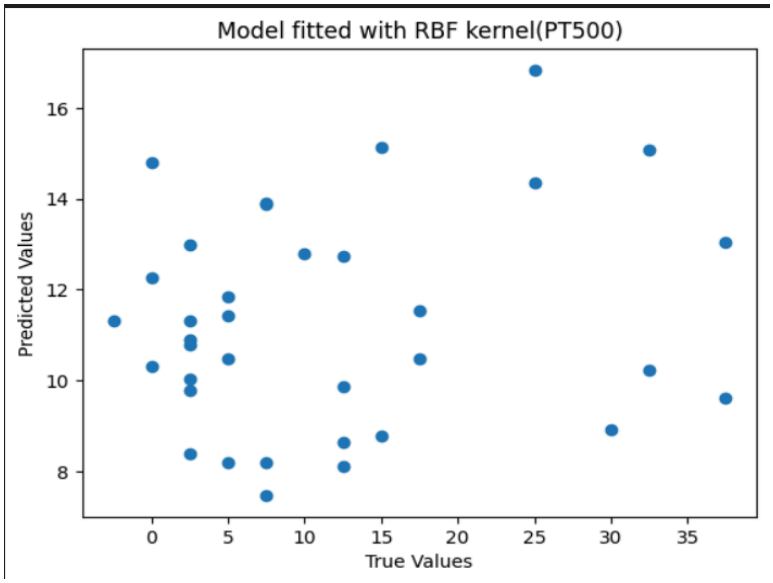


MODEL WITH LINEAR KERNEL(PT4000):

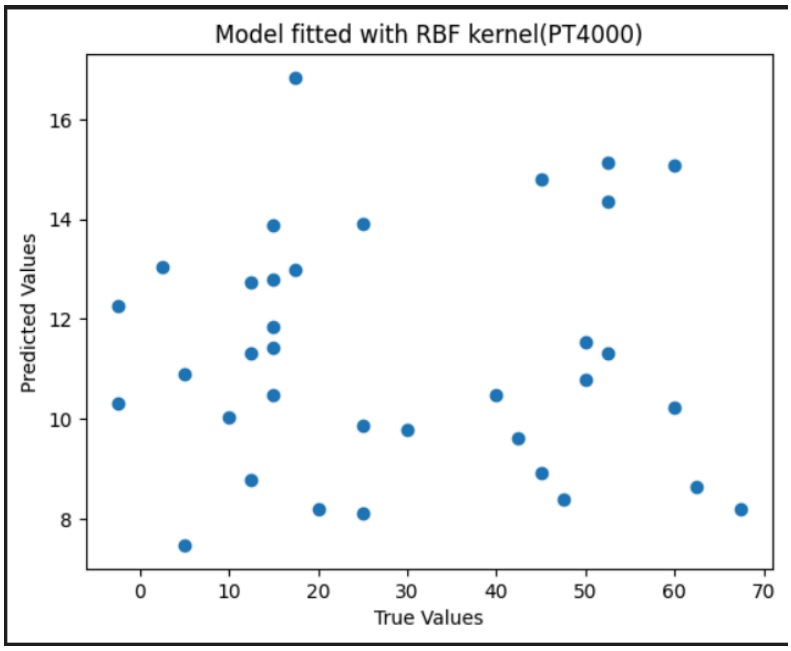


RBF Kernel:

MODEL WITH RBF KERNEL (PT500):



MODEL WITH RBF KERNEL(PT4000):



MODEL EVALUATION METRICS:

Linear Kernel	PT500	PT4000
R2 score	-0.012	-0.83149
MAE	8.920	21.21515
MSE	128.227	764.7036

RBF Kernel	PT500	PT4000
R2 score	0.0258	-0.77
MAE	8.815	20.471
MSE	123.431	742.199

Based on the aforementioned criteria, it is evident from the results that support vector regression outperformed linear kernels with an RBF kernel. Compared to a linear kernel, RBF kernel captures complex features better. Values generated by the SVR model with the linear kernel are obviously inferior to the R2 scores of the PT500 and PT4000 values of the SVR model with "RBF."

3. Convolutional Neural Network:

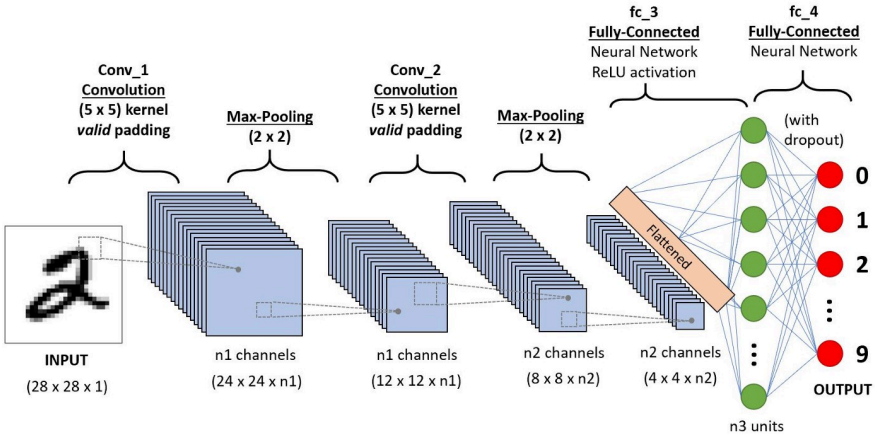
Introduction:

Convolutional Neural Networks (CNNs) have revolutionized various fields of pattern recognition, such as speech and image processing. One of their key advantages lies in their ability to reduce the parameters in an Artificial Neural Network (ANN), which allows for handling more complex tasks with larger models compared to traditional ANNs. Additionally, CNNs excel at extracting abstract features as data moves through deeper layers. They're designed with the assumption that their features should be spatially independent. By incorporating both hearing data and grey matter images as input channels, CNNs can effectively fuse multiple sources of information for predictive tasks. [7]

Application: Convolutional Neural Networks(CNNs) are well-suited for processing spatial data such as gray matter images from MRI scans. They can automatically learn hierarchical representations of features from raw input data, potentially capturing intricate patterns and relationships within the images. Multimodal Learning: CNNs can integrate both hearing data and gray matter images as input channels, allowing for the joint analysis of multiple modalities of information.

Feature Learning: By learning features directly from the data, Convolutional Neural Networks (CNNs) do away with the requirement for human feature engineering. This has the potential to be useful in capturing intricate linkages within the data.

Complexity and Interpretability: Convolutional Neural Networks are more complex models compared to linear regression and SVM, and they may require more computational resources for training. Additionally, interpreting the learned features in CNNs can be challenging.



Convolutional Neural Network [5]

We had 13 convolutional layers in the model.

Filters: Kernels, another name for filters, are tiny matrices with weights (such 3x3 or 5x5). Throughout the training process, these weights are memorized. Every filter is in charge of identifying a particular pattern or characteristic—such as edges, textures, or shapes—in the input image. Filters with 32,64,128,256,512,1024 and 2048.

Convolutional Layer: Convolutional neural networks (CNNs), often applied in image processing tasks within deep learning, are structured with key elements known as convolutional layers. These layers utilize convolution techniques to extract features from the input data.

The neural network can learn hierarchical representations of the input data with the aid of convolutional layers. While subsequent layers combine these information to recognize more complicated patterns, like objects or faces, the initial few layers may only be able to detect fundamental properties like edges and corners. [8]

Pooling Layer: Pooling layers offer a standard downsampling technique, decreasing the in-plane dimensions of feature maps. This introduces translation invariance to minor shifts and distortions while reducing the number of learnable parameters in subsequent layers. Unlike filter size, stride, and padding, which are hyperparameters in pooling processes similar to convolution, it's important to note that pooling layers do not possess any learnable parameters.[8]

In order to decrease the spatial dimensions and downsample the feature maps, pooling layers

are frequently included. Pooling lowers computing complexity and increases the learned features' invariance to slight modifications.

In our model, we added a pooling layer to reduce the dimension of the data.

Feed Forward Layer: Feed-forward neural networks are the most well-liked and often utilized models. They go by a variety of names, including "multilayer perceptrons" (MLP). An approach for classification with biological inspiration is a feed-forward neural network. It is made up of several basic processing units that resemble neurons that are arranged in layers, with connections between all of the units in the layer above and below. Each of these connections may have a distinct weight or strength, therefore they are not all created equal. The knowledge of a network is encoded by the weights on these links. Neural network units are commonly referred to as nodes.

Layer by layer, data enters at the input and travels through the network to reach the output. The network's inputs alone make up the input layer. A hidden layer, made up of any number of neurons or hidden units arranged in parallel, comes next. The weighted summation of the inputs is carried out by each neuron, after which the neuron function—also known as the transfer/activation function—is passed. [6]

- Once the input data has gone through the convolutional layers, it goes through a feed forward layer to output to classes.

Training Parameters:

Epoch: When a neural network is being trained, an epoch is one full run of the training dataset. The model views the complete dataset during each epoch, computes the loss, and modifies its parameters (weights and biases) in accordance with the optimization procedure.Every epoch modifies the model's parameters to enhance its performance on the job it is being trained for, such picture classification, allowing the model to learn from the data.

In our model, we trained for 10 epochs.

Adam Optimizer:

The term "Adam" optimizer stands for Adaptive Moment Estimation, representing an iterative optimization process used to minimize the loss function during neural network training.

Our model used the Adam optimizer.

Loss Function: A loss function, also referred to as a cost function, evaluates the correspondence between the predictions generated by the network through forward propagation and the ground truth labels. Cross-entropy is frequently utilized in multiclass classification scenarios, whereas mean squared error is more common in regression tasks involving continuous data. The choice of loss function is a critical hyperparameter that should be selected based on the specific objectives of the task.

A metric called cross-entropy loss is used to assess how well a classification model forecasts the likelihood of various classes for every input. It computes the discrepancy between the classes' actual distribution and the anticipated probability distribution. Put another way, it indicates how "wrong" the model's predictions are in relation to the actual results.Lesser the cross entropy loss, more would be the accuracy of model regression. [8]

In the model that we measured loss for predicting threshold values based on different frequencies Pt500 and Pt4000 Hz, we utilized cross entropy loss.

Overfitting: Overfitting happens when a model learns intricate patterns present only in the training data, essentially memorizing noise rather than the underlying signal. To automatically detect overfitting, one can track accuracy and loss on both training and validation sets. If the model performs well on the training data but poorly on the validation data, it's likely overfitting. Various strategies exist to combat overfitting, with acquiring more training data being the most effective solution.[8]

- We shuffled the data to make sure the network did not learn the order of the data and to add some randomness in.

Model Layers:

PT500-

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d (Conv3D)             (None, 58, 58, 28, 32)    896

 conv3d_1 (Conv3D)           (None, 56, 56, 26, 32)    27680

 conv3d_2 (Conv3D)           (None, 54, 54, 24, 64)    55360

 conv3d_3 (Conv3D)           (None, 52, 52, 22, 64)    110656

 conv3d_4 (Conv3D)           (None, 50, 50, 20, 128)   221312

 conv3d_5 (Conv3D)           (None, 48, 48, 18, 128)   442496

 conv3d_6 (Conv3D)           (None, 46, 46, 16, 256)   884992

 conv3d_7 (Conv3D)           (None, 44, 44, 14, 256)   1769728

 conv3d_8 (Conv3D)           (None, 42, 42, 12, 512)   3539456

 conv3d_9 (Conv3D)           (None, 40, 40, 10, 512)   7078400

 conv3d_10 (Conv3D)          (None, 38, 38, 8, 1024)   14156800

 conv3d_11 (Conv3D)          (None, 36, 36, 6, 1024)   28312576

 conv3d_12 (Conv3D)          (None, 34, 34, 4, 2048)   56625152

 max_pooling3d (MaxPooling3  (None, 17, 17, 2, 2048)   0
 D)
```

```
 max_pooling3d_1 (MaxPoolin  (None, 8, 8, 1, 2048)     0
 g3D)

 flatten (Flatten)           (None, 131072)            0

 dense (Dense)               (None, 1024)              134218752

 dropout (Dropout)           (None, 1024)              0

 dense_1 (Dense)             (None, 512)               524800

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 1)                 513

=================================================================
Total params: 247969569 (945.93 MB)
Trainable params: 247969569 (945.93 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

PT4000-

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d (Conv3D)             (None, 58, 58, 28, 32)    896

 conv3d_1 (Conv3D)           (None, 56, 56, 26, 32)    27680

 conv3d_2 (Conv3D)           (None, 54, 54, 24, 64)    55360

 conv3d_3 (Conv3D)           (None, 52, 52, 22, 64)    110656

 conv3d_4 (Conv3D)           (None, 50, 50, 20, 128)   221312

 conv3d_5 (Conv3D)           (None, 48, 48, 18, 128)   442496

 conv3d_6 (Conv3D)           (None, 46, 46, 16, 256)   884992

 conv3d_7 (Conv3D)           (None, 44, 44, 14, 256)   1769728

 conv3d_8 (Conv3D)           (None, 42, 42, 12, 512)   3539456

 conv3d_9 (Conv3D)           (None, 40, 40, 10, 512)   7078400

 conv3d_10 (Conv3D)          (None, 38, 38, 8, 1024)   14156800

 conv3d_11 (Conv3D)          (None, 36, 36, 6, 1024)   28312576

 conv3d_12 (Conv3D)          (None, 34, 34, 4, 2048)   56625152

 max_pooling3d (MaxPooling3  (None, 17, 17, 2, 2048)   0
 D)
```

```
 max_pooling3d_1 (MaxPoolin  (None, 8, 8, 1, 2048)     0
 g3D)

 flatten (Flatten)           (None, 131072)            0

 dense (Dense)               (None, 1024)              134218752

 dropout (Dropout)           (None, 1024)              0

 dense_1 (Dense)             (None, 512)               524800

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 1)                 513

=================================================================
Total params: 247969569 (945.93 MB)
Trainable params: 247969569 (945.93 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Results:

PT500-

```
Epoch 1/10
7/7 [==============================] - 1416s 113s/step - loss: 46.8275 - accuracy: 0.7491 - val_loss: 6.3715 - val_accuracy: 0.9804
Epoch 2/10
7/7 [==============================] - 178s 26s/step - loss: 2.3280 - accuracy: 0.9715 - val_loss: 0.0202 - val_accuracy: 0.9804
Epoch 3/10
7/7 [==============================] - 183s 26s/step - loss: 0.0445 - accuracy: 0.9804 - val_loss: 0.0200 - val_accuracy: 0.9804
Epoch 4/10
7/7 [==============================] - 183s 26s/step - loss: 0.2248 - accuracy: 0.8825 - val_loss: 0.0193 - val_accuracy: 0.9804
Epoch 5/10
7/7 [==============================] - 182s 26s/step - loss: 9.4195 - accuracy: 0.9003 - val_loss: 0.0348 - val_accuracy: 0.9804
Epoch 6/10
7/7 [==============================] - 175s 25s/step - loss: 0.0595 - accuracy: 0.9804 - val_loss: 0.0473 - val_accuracy: 0.9804
Epoch 7/10
7/7 [==============================] - 181s 26s/step - loss: 0.0444 - accuracy: 0.9715 - val_loss: 0.0333 - val_accuracy: 0.9804
Epoch 8/10
7/7 [==============================] - 174s 25s/step - loss: 0.0255 - accuracy: 0.9804 - val_loss: 0.0192 - val_accuracy: 0.9804
Epoch 9/10
7/7 [==============================] - 174s 25s/step - loss: 0.0224 - accuracy: 0.9804 - val_loss: 0.0262 - val_accuracy: 0.9804
Epoch 10/10
7/7 [==============================] - 174s 25s/step - loss: 0.0210 - accuracy: 0.9804 - val_loss: 0.0197 - val_accuracy: 0.9804
```

PT4000-

```
Epoch 1/10
7/7 [==============================] - 1368s 109s/step - loss: 0.4278 - accuracy: 0.8434 - val_loss: 0.0864 - val_accuracy: 0.9880
Epoch 2/10
7/7 [==============================] - 180s 26s/step - loss: 22.2014 - accuracy: 0.7440 - val_loss: 0.0126 - val_accuracy: 0.9880
Epoch 3/10
7/7 [==============================] - 171s 25s/step - loss: 0.0774 - accuracy: 0.9880 - val_loss: 0.0146 - val_accuracy: 0.9880
Epoch 4/10
7/7 [==============================] - 177s 26s/step - loss: 0.0162 - accuracy: 0.9880 - val_loss: 0.0119 - val_accuracy: 0.9880
Epoch 5/10
7/7 [==============================] - 177s 26s/step - loss: 0.0131 - accuracy: 0.9880 - val_loss: 0.0119 - val_accuracy: 0.9880
Epoch 6/10
7/7 [==============================] - 176s 25s/step - loss: 0.0126 - accuracy: 0.9880 - val_loss: 0.0119 - val_accuracy: 0.9880
Epoch 7/10
7/7 [==============================] - 176s 25s/step - loss: 0.0122 - accuracy: 0.9880 - val_loss: 0.0119 - val_accuracy: 0.9880
Epoch 8/10
7/7 [==============================] - 175s 25s/step - loss: 0.0122 - accuracy: 0.9880 - val_loss: 0.0119 - val_accuracy: 0.9880
Epoch 9/10
7/7 [==============================] - 175s 25s/step - loss: 0.0120 - accuracy: 0.9880 - val_loss: 0.0119 - val_accuracy: 0.9880
Epoch 10/10
7/7 [==============================] - 167s 24s/step - loss: 0.0120 - accuracy: 0.9880 - val_loss: 0.0119 - val_accuracy: 0.9880
```

	PT500	PT4000
Mean Training MSE	5.901	2.27
Mean Validation MSE	0.661	0.019

Lessons Learned

We have gained a great deal of knowledge in the process of developing this model to categorize an MRI brain scan. In order to construct a dataset that could be fed into a neural network, we first learnt how to investigate existing datasets. The data from the.nii files had to be extracted, and each MRI's correct label had to be mapped to it before we could generate our input dataset. We discovered how to successfully subsample the data throughout the data extraction process to avoid computational problems while using an excessively huge image. We discovered how crucial it is to understand the data that one is using during the dataset development process. Lacking in-depth understanding of the data being used, the performance of the task that is being conducted might not be as great as it could be.

Another lesson we also learned how effective convolutional neural networks are in classifying images. We were really aback by how strong and efficient these neural networks could be after only a little training period. These models have countless applications in research, and neuroscience experts can save a lot of time by using a highly trained convolutional neural network to determine whether an MRI image is personally recognizable.

Finally, we discovered how crucial it is to select the right hyperparameters and model architecture. By adding in dropout layers, we reduced the chance of overfitting our training data. We certainly learned the importance of choosing appropriate architecture and hyperparameters.

Conclusion

Linear regression, SVM and CNN are popular regression models. Linear regression and SVM are generally used in regression and classification tasks respectively. In this study, linear regression, SVM, and CNN models were employed to predict hearing thresholds at different frequencies using gray matter images and associated thresholds data. Each model's threshold values at PT500 and PT4000 were considered separately for training and testing purposes. This approach ensured that the models were trained and evaluated independently for each pure-tone frequency, allowing for a more focused analysis of their performance.Various evaluation metrics such as R-squared score, Mean Squared Error (MSE), and Mean Absolute Error (MAE) were utilized to determine the best-performing model among the three.CNN has the lowest mean squared error value for both Pt500 and Pt4000 values, which are 5.9 and 2.9, respectively,

according to the assessment metrics that have been observed. These numbers are determined by training loss. SVR definitely performed better than linear regression. Out of the three models, CNN does the best when it comes to the input data.

References

[1]Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2024, March 5). *Understanding of convolutional neural network*. IEEE Explore. Retrieved April 19, 2024, from https://ieeexplore.ieee.org/document/8308186

[2]Dharmaraj. (2022, June 1). *Convolutional Neural Networks (CNN) — Architecture Explained | by Dharmaraj*. Medium. Retrieved April 19, 2024, from https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243

[3]Georgia, E. I., & Tigas, S. K. (2010). *Feedforward Neural Network*. Science Direct. Retrieved April 19, 2024, from https://www.sciencedirect.com/topics/chemical-engineering/feedforward-neural-network/

[4]Jain, S. (2024, March 20). *Linear Regression in Machine learning*. GeeksforGeeks. Retrieved April 19, 2024, from https://www.geeksforgeeks.org/ml-linear-regression/

[5]Kanade, V. (2022, September 29). *All You Need to Know About Support Vector Machines*. Spiceworks. Retrieved April 19, 2024, from https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/

[6]Keyvanpour, M. R., & Shirzad, M. B. (2024, March 5). *Support Vector Regression*. Science Direct. Retrieved April 19, 2024, from https://www.sciencedirect.com/topics/computer-science/support-vector-regression

[7]Niemeyer, S. (2014, June 3). *How to Read an Audiogram and Determine Degrees of Hearing Loss*. The National Hearing Test. Retrieved April 19, 2024, from https://www.nationalhearingtest.org/wordpress/?p=786

[8]Schneider, A., Hommel, G., & Blettner, M. (2010, Nov 5). *Linear Regression Analysis - PMC*. NCBI. Retrieved April 19, 2024, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2992018/

[9]Yamashita, R., Nishio, M., Do, R., & Togashi, K. (2018, June 22). *Convolutional neural networks: an overview and application in radiology - Insights into Imaging*. Insights into Imaging. Retrieved April 19, 2024, from https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

[10]Buyukyildiz, M., Kahramanli, H., & Tezel, G. (2013). *Application of Support Vector Regression (SVR) for monthly evaporation prediction*. https://www.semanticscholar.org/paper/Application-of-Support-Vector-Regression-(SVR)-for-Buyukyildiz-Kahramanli/700a36f0108409a47651f9a99a672b711d358d2b

[11] Akay, Ö., & Tunçeli, M. (2021). Use of the Support Vector Regression in Medical Data Analysis. *Experimental and Applied Medical Science*, *2*, 242–256. https://doi.org/10.46871/eams.2021.29

[12] Renukadevi, N., & Thangaraj, P. (2013). Performance Evaluation of SVM -RBF Kernel for Medical Image Classification Performance Evaluation of SVM RBF Kernel for Medical Image Classification Performance Evaluation of SVM -RBF Kernel for Medical Image Classification. *Global Journal of Computer Science and Technology Graphics & Vision*, *13*. https://globaljournals.org/GJCST_Volume13/3-Performance-Evaluation.pdf