Speech Engine: Demonstrating the Impact of Pseudo-Labeling and Autoencoder on Downstream Classification Tasks

Harish Ram*, **Jiachen Sands***, **Sisi Zhang***, **Zeqiu Zhang***, **Amir Jafari**
hram@gwu.edu, jli0@gwu.edu, szhang61@gwu.edu, zzhang77@gwu.edu, ajafari@gwu.edu

The George Washington University

Abstract

Our study on speech classification shows the results of several types of convolutional neural networks and transformer models with the use of pseudo labeling, augmentation and an autoencoder for pre-training to increase model performance. Since the human aspect of the data must remain the central focus of analysis, allowing users to classify their speech requires a certain degree of data manipulation to not distort the quality of data. As a result, we designed a Speech Engine to classify emotions, race, accent, age, sex and Parkinson's disease from raw audio files with high accuracy.

1 Introduction

Audio analysis has many facets in different industries ranging from audio forensics at crime scenes or home security systems, AI listening devices such as Google Home, Amazon Alexa or Siri or even interpreting phone conversations for quality assurance purposes. Processing audio data into an understandable format is a relatively new form of data analysis that is much more complex than tabular or text data. In this study, we will be explaining the process of building a Speech Engine that will allow anyone to record their audio to upload resulting in classification based on their emotion, sex, age, race, accent and the presence of Parkinson's disease.

With the diverse set of data that we have collected from multiple voice studies that contain participants from numerous countries, backgrounds and ethnicities, we can extract a higher number of features within each category such as more types of emotions and a wider range of age groups to make the speech engine more applicable to more users. The primary obstacle is preprocessing the data into a viable format for training machine learning models all the while maintaining the integrity of the data; therefore, it is necessary to maintain a balanced class size throughout the project, so the results will not be biased. With a topic like speech classification, when manipulating the data to increase model performance, we must always make sure to not distort the audio too much so that it would be unrecognizable from the original data.

We have designed a speech engine to let users input raw audio files and output classification results and the code can be found on GitHub repository[1]. We decided to use raw audio files in .wav and .mp3 format to then convert them into Mel Spectrograms to accomplish different classification tasks. For different categories, we used data augmentation to make sure the number of class labels was balanced. After preprocessing, we were able to get a reasonable baseline score from our 3-layer CNN model, which we then used as a benchmark to compare to other pre-trained models and even the state-of-the-art transformer, Wav2Vec 2.0 which has a significantly more complex model architecture.

2 Problem Statement

The two largest challenges for this project can be categorized into data manipulation and model development. Several unique datasets were used for different categories; thus, processing can be a complex obstacle when dealing with data that is not already standardized.

For Parkinson's disease and accent data, the biggest issue was that the audio files were too large to process, so it was necessary to split them into smaller segments and remove the silences where required. However, this poses another complication during training and testing because the model needs to be tested on the original, unsegmented audio files to output an F1 score. If the audio that is trained on the model is segmented, then the model

*Co-first authors, equal contribution

[1] https://github.com/sz389/Capstone-Group5

will only learn from small snippets of audio that do not reflect real-life speech.

Furthermore, the data is imbalanced for some categories, which is the primary setback for classification problems. If a speech engine is trained on unbalanced audio data then regardless of the user's voice, the classification will be biased towards the majority class of the data. Similarly, even if the dataset is balanced, yet does not contain enough data, the model will still suffer in production as it will not be trained on diverse enough data to understand more distinct types of voices.

The model parameters for different categories can be different, so tuning them properly can produce significantly better results. This needs to be done differently for each category. Some tasks such as emotion are much more complicated for the model to learn from due to the slight discrepancies between one emotion to the other compared to tasks such as sex where the pitch is the dominant differing factor. Therefore, when testing the performance of models, it is important for all parameters to be tuned individually and tested thoroughly.

On the same note, generating Mel Spectrograms is fundamental even for a study on sound data. Since spectrograms are a visual representation of audio, they can be extremely useful in training convolutional neural networks because of their high dimensionality in feature extraction. The problem is that generating Mel Spectrograms can not only pose a time issue in that creating an image representation of thousands of audio files can take hours, but also that they have their own set of parameters that need to be tuned for each classification task separately.

Audio data analysis is a complicated field due to the nature of the data. Common pre-training and augmentation techniques can sometimes have a neutral or negative effect on performance simply due to how difficult it is to learn from audio and image data. As a result, using only augmentation or only a few extra layers in a CNN model will occasionally produce worse results because of how complex the data is. It is necessary to try several techniques or a combination of methods for an individual classification task to increase the chance of improving model performance.

3 Related Work

In recent years, convolutional neural networks (CNN) have started to become a base architecture for data scientists to start modeling with (13)(21). So we started with a 3-layer CNN model from George Washington University's professor, Amir Jafari, as a basic model for CNN. Using the MNIST dataset, the author constructed a rudimentary CNN model. Our team has added extra convolution layers to 9 and 11 layers as beginning points to improve model accuracy. In addition, we used pre-trained models such as ResNet18, ResNet34, VGG16, and EfficientNet_b2 to evaluate performance in each category.

Referencing some studies(19)(18)containing common augmentation techniques for audio data, we built data augmentation code to handle data imbalance issues prior to the CNN implementation. We attempted a variety of augmentation approaches in this section of the code, such as adding white noise, time shift, and pitch scale to the raw audio files and then generating extra Mel Spectrograms. Finally, we mixed the Mel Spectrograms from the original audio files with the newly created Mel Spectrograms and used them on the best CNN model.

Our understanding of the autoencoder(12) and its functionality comes from a study(7)using the MNIST(6) dataset to pass the images through an autoencoder. The purpose was to determine how an encoder and decoder work to learn the latent space representations of the features extracted from the image and reproduce a similar image. Since the study recommended using convolutional layers as the architecture in the autoencoder in the future, that was the motivation for unsupervised learning with the autoencoder.

Some studies(8)(11) found that pseudo-labeling can help to increase the model performance by training on unlabeled data when there is limited labeled data available. In one study. pseudo-labeling(16)is proven to be useful on the MNIST dataset by using only 1,000 data points of labeled data and using the rest of the dataset containing 59,000 data points as unlabeled data. Using semi-supervised learning, this project was able to output better results using pseudo-labeling than training on only 1,000 points of labeled data. The one point of concern in this study is the unlabeled data that was used was manually collected from specific classes, so there would be an even distribution of data. This is typically not the case in real-life pre-training situations where the only factors that can be controlled are the subject matter and size of the data. Therefore, the main point we took away from this study was the concept of semi-supervised learning which proved to be useful in our project. Furthermore, the Wav2Vec 2.0 paper(4) was incredibly useful in our study as it posed the issue of attaining purely labeled data and proves that pretraining on unlabeled data with a framework of self-supervised learning can outperform models that use 100 times more labeled data. Understanding the idea that speech recognition models can be as powerful with-

out annotated data, we set up a semi-supervised learning approach for pretraining by using a similar amount of unlabeled data as our original labeled data.

4 Methodology

4.1 Augmentation

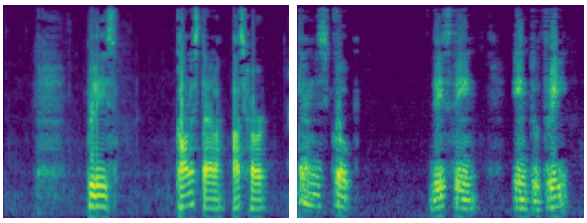

Figure 1: Before *(Left)* and After *(right)* Augmentation

For different classification tasks in our project (emotions, age, accents, etc.), different augmentation methods yielded the best performance. In all these augmentation techniques, there are corresponding parameters that can be changed as well. For example, with time stretch, adjusting the factor will change to what degree the audio speed is stretched. Figure 1 shows how the Mel Spectrograms look before (left) and after (right) time stretch is applied. Other augmentation techniques listed work in similar ways by altering the values in the audio signal array through non-linear calculations.

4.2 CNN

In Figure 2, a Convolutional Neural Network (CNN)(3) consists of several types of layers that all perform a specific type of feature extraction from the input image, which in this project, is a Mel Spectrogram. CNN architectures work best for Mel Spectrograms as opposed to a basic MLP (Multilayer Perceptron) because CNN can capture the spatial and temporal dependencies of the pixels in the image. Since Mel Spectrograms are RGB images meaning they contain multiple channels, the concept of pixels depending on adjacent pixels is important. Moving on to the architecture of CNN, the convolutional layer which comprises the kernel and stride. The kernel is a matrix of reduced size compared to the input image that traverses the entire image and performs matrix multiplication in the portions it occupies each stride to extract a representation of the image in a one channel convolutional feature output to pass along to the next convolutional layer. In terms of the actual size of the image (height and width), we use the same padding feature to maintain the same dimensionality throughout the network. To allow more layers in the CNN architecture to reduce the computational power required by the model, pooling layers are necessary to take the maximum value of each portion of the image the kernel covers. Finally, a fully connected layer is a linear layer to make sure that the dimension of output is the same as the number of labels. The activation functions in the form of the activation function ReLU to perform gradient calculation as defined by f(x) = max(0, x) to speed up training.

4.3 Autoencoder

An autoencoder is a neural network that compresses the input (an image) into a downgraded representation and reconstructs the image as the output. In Figure 3, the encoder is where the original input, which in our case is a Mel Spectrogram, is compressed and produces an encoded representation of the image. The decoder then uses this representation to reconstruct the image as the same size and parameters as the input.

$$H_{out} = \left\lfloor \frac{H_{in}+2\times pad[0]-d[0]\times(k[0]-1)-1}{stride[0]} + 1 \right\rfloor$$

$$\mathrm{W}_{out} = \left\lfloor \frac{W_{in}+2\times pad[0]-d[0]\times(k[0]-1)-1}{stride[0]} + 1 \right\rfloor$$

To determine the exact dimensions of the inputs (number of channels and image size) as the tensor is passed through each layer, we used the equations above(14) to calculate the image size, pooling, dilation (d), kernel size (k), padding of each layer to make sure the tensors passed through each layer were properly processed through the encoder. This equation comes from the Conv2d documentation using the Pytorch framework and was used for our calculations as we wanted to apply a 2-dimensional convolution over our input which was composed of several channels.

$$\mathrm{H}_{out} = (H_{in}-1)\times stride[0] - 2\times pad[0] + d[0]\times(k[0]-1) + pad_{output}[0] + 1$$

$$\mathrm{W}_{out} = (W_{in}-1)\times stride[0] - 2\times pad[0] + d[0]\times(k[0]-1) + pad_{output}[0] + 1$$

For the decoder, we calculated the input dimensions across each layer using the equations above (15)since the ConvTranspose2d layers comprise our decoder. However, this new image will not be exactly the same as the input. For this project, it is important that the colors of the Mel Spectrograms are carefully analyzed by the model as they are representations of the actual audio. The purpose of using an autoencoder, for

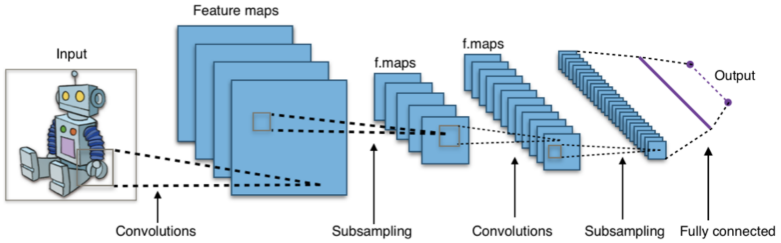

Figure 2: CNN Architecture(20)

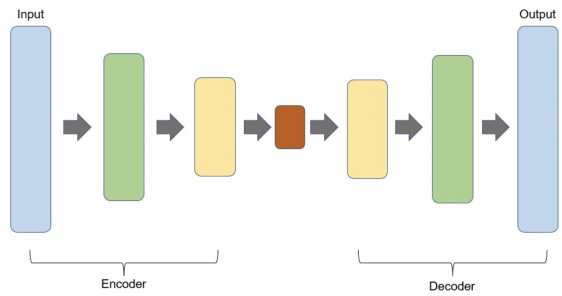

Figure 3: Architecture of Autoencoder

this project, is to pre-train our data in order to use the weights and biases as a starting point for a CNN model. Specifically, we pre-trained the autoencoder with both the encoder and decoder; however, only saved the encoder model parameters. By doing this, we can load the model from the autoencoder before beginning training on CNN. This will ideally increase the performance of the 3-layer CNN model.

4.4 Pseudo-labeling

Since data augmentation failed to improve the model accuracy further and we have not yet used an autoencoder for pre-training with a more complex architecture, we also tried pseudo-labeling using semi-supervised learning. The main idea behind pseudo-labeling is to improve model accuracy by training labeled and unlabeled data simultaneously. The benefit of pseudo-labeling is seen mainly when there is a lack of labeled data available entirely or for specific minority classes. Ideally, the unlabeled data should have a balanced distribution; however, this is impossible to tell as the data is unlabeled. Furthermore, the subject matter of the data should be similar to the labeled data as well. For example, if the goal is to improve model performance using race classification, then the unlabeled data should contain speakers with a diverse distribution of races.

$$Loss = Loss_{labeled} + \alpha(t) \times Loss_{unlabeled}$$

The following four steps will explain this idea in more detail. First, we have trained multiple CNNs, pre-trained models, and saved the best model that has the highest accuracy for each category. Next, using the parameters from the best model, the labels that are predicted on the unlabeled data become pseudo-labels. Then, these pseudo labels are used while training the model on the unlabeled data when the unlabeled loss is calculated. In every epoch, the model is updated by learning from backpropagation using the loss calculated from the pseudo labels. However, within each epoch, the labeled data is also trained on the model every several batches of unlabeled data. By using this semi-supervised learning technique, the model uses both the labeled and unlabeled loss to perform backpropagation to allow the model to learn from both labeled and unlabeled data.

There are three important hyperparameters in this process that heavily influence the performance of training. Alpha is used to control the weight applied to the unlabeled loss. Essentially, this determines how much influence the unlabeled loss has in backpropagation. T_1 refers to the number of steps used to train on labeled data from the previous supervised training; while, T_2 is the number of steps until alpha reaches its maximum value. In the formula below(16), $\alpha(t)$ shows how weight changes over time.

$$\alpha(t) = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1}\alpha_f & T_1 \leq t < T_2 \\ \alpha_f & T_2 \leq t \end{cases}$$

When t $< T_1$, this illustrates the idea of when the model is trained purely on labeled data as we need to load in the model weights of the supervised training model right before the pseudo-labeling training process. Thus, during semi-supervised

training, since T_1 and t will begin at the same value, the alpha will always be greater than 0. Depending on the number of epochs and steps that have passed during training, the alpha will be determined by the next equation where t is between T_1 and T_2. This value will begin at a near-zero number and slowly increase towards the maximum value of alpha which we have defined as 0.5 because we do not have high confidence in our unlabeled data; therefore, we do not want the unlabeled loss to be weighted too highly during backpropagation. Finally, once t is greater than T_2 which generally occurs in the last several epochs of training, the alpha will stabilize to 0.5.

In summary, before reaching T_2 steps, the influence of unlabeled data is slowly increasing from 0. In other words, the unlabeled data consistently become more effective in the training process as the number of steps increases to T_2.

5 Results

The results that we have collected throughout this project reflect the level of complexity that data processing in the speech recognition field requires. In this section, we will go through the metrics have yielded, explain the reasoning behind these scores and how they can be improved with future work.

5.1 Experimentation Protocol

The process of making better predictions in terms of speech classification begins with formatting the data in such a way as to yield accurate results. To determine an acceptable benchmark performance, we decided to use a 3-layer CNN model. We first experimented with using solely linear layers which did not prove very accurate as our input - Mel Spectrograms - contained 3 channels (RGB). The preliminary step was to tune the parameters of the Mel Spectrograms to produce the most well-representative images. Thus, we explored using convolutional, pooling, activation and fully connected layers as these have been noted to work more effectively for images with color and pixel complexity. The issue with this approach was how the number of layers to use before accuracy began diminishing. On account of there being several classification tasks that we were observing, we tested using 3, 5, 7, 9 and 11 convolutional layers which proved varying results for different tasks.

After demonstrating the capability of our own CNN model, we used pre-trained models - ResNet18, ResNet34, VGG16 and EfficientNet b2 to train the data and compare results. These models have more layers. Similarly, different tasks produced different results. We would tune the parameters of the model to achieve better results in all downstream tasks.

Our next approach involved building an autoencoder model to pre-train our data first on an encoder and decoder, save the model weights and biases of the encoder and begin training with the autoencoder weights and biases as the starting point. The objective of doing this was to use an autoencoder to extract features from spectrograms with reduced dimensions to allow for higher performance with a subsequent model matching the same architecture of the encoder. The architecture of the decoder was the exact opposite of the encoder in that we re-built the encoded image to the original size and number of channels to calculate loss against the original input. Using unsupervised learning with the autoencoder allowed us to save model parameters that were pre-trained on our data, so we could use them in a supervised learning model such as CNN.

The results of our autoencoder did not yield the degree of performance we wanted in pre-training, so we used another pre-training method with semi-supervised learning that would generate pseudo-labels of unlabeled data, so we can increase our sample size and potentially increase the model performance. We performed preliminary experimentation on the MNIST dataset as a proof of concept by using different sizes of unlabeled data and comparing it to the original score of training on the full dataset. In both cases, the test set was untouched. The results of this study showed that pseudo-labeling could offer similar performance where of the data was unlabeled and of the data was labeled compared to model F1 where all of the data was labeled. Based on these results, we decided to approach pseudo-labeling with the idea that additional unlabeled data will not be a detriment to performance.

For the purpose of this study, the unlabeled data used for the emotions, race, sex and age classification task was the accent data without labels and vice versa for the accent classification task. The disease classification task used data from a recording from the LJ speech dataset(9). We performed pseudo-labeling using the models that previously yielded the best results. However, again, our results were varying. As a result, we have decided to simply use the models that have yielded the best results to build the speech engine. To do this, we have loaded the model parameters for each category into the GUI code file and performed prediction after the user uploads an audio file or speaks into the microphone. The measurements that we primarily observed were the F1 score and confusion matrix as our project revolves around accurate classification.

5.2 Data Description

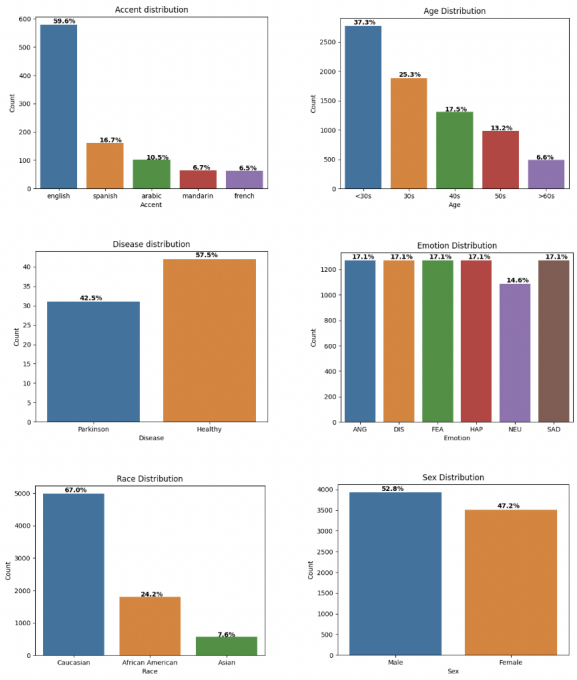

Figure 4: Distribution of Data

In this study, we have used data from four sources to perform multiple classification tasks. For the accent category, the top 5 most frequent native languages were taken from a Kaggle dataset(17). Before splitting the audio data into 5-second segments, the size of the dataset was 971 then 5,566 after being split. The downstream tasks of race, sex, emotion, and age classification share the same dataset(5) with 7,360 samples after dropping NA values. For the Parkinson's Disease category, the data comes from Zenodo (10). There are 73 samples in this dataset and 1,280 samples after splitting it into chunks for loading-efficiency purposes which was also done for the accent category as well.

5.3 Data Tables

Table 1 shows the F1 score of each category's performance starting with the benchmark score using a 3-layer CNN followed by using the autoencoder for pre-training. Then, the 'Best CNN*' column refers to the best scores generated by all the pre-trained models including CNN9 , Renset18, ResNet34 which we then performed Augmentation on. The last two columns represent the best CNN model using pseudo-labeling and finally Wav2Vec2.

To demonstrate the difference in performance between models, it is critical to establish a baseline score. We used a 3-layer CNN model to institute a benchmark because it is a basic, yet sophisticated model that was still properly tuned for each category. The number of labels correlates to the Accuracy and F1 score meaning that more labels correspond to a lower baseline score because it is more difficult to predict more classes. Therefore, Sex has the highest baseline greater than 90% understandably as the difference between a male and female voice is more distinct than various age groups or unique emotions. Throughout this study, we recorded both the accuracy and F1 score for all of our models; however, because this study is exclusively about classification, the F1 score is more meaningful as it takes into account precision and recall.

The F1 scores of the 3-layer CNN model for all categories are shown in Table 1 before and after the autoencoder implementation. The architecture of the Encoder must match the architecture of the model used to perform classification which is why the 3-layer CNN Benchmark is used to analyze the performance of the autoencoder. This is because once the autoencoder is trained on audio training data, the model weights and biases are loaded into a 3-layer. Except for the accent task, the F1 scores in all categories have improved marginally. The reason we used 3 layers in the autoencoder as opposed to more layers was simply for proof of concept which turned out to be a success; however, adding more layers to the CNN model has the potential to improve model performance which is something we plan to do in the future.

Furthermore, looking at the 'Best CNN*', compared to the benchmark, the CNN9 model is better for disease by 22% and emotions by 8%, ResNet18 is better for age by 31% and sex categories by 5%; while, ResNet34 is better for accent by only 2% and race categories by 17%. The models listed in the table were the best for our data out of all the models we tested which include a 3 and 9 layer CNN, ResNet18, ResNet34, VGG16 and EfficientNet_b2. It is also important to note that different types of audio data can expect better results from different models, regardless of the complexity of the architecture. Overall, in relation to the benchmark, all categories yielded a noticeable increase.

Table 1 also shows the F1 score before and after data augmentation on the best model for each category. We applied augmentation on raw audio files for imbalanced tasks which include accent, age and race. We tried different combinations of augmentation methods and recorded the best score. After all the experiments, we found that augmentation did not working well for accent and age categories and only made small progress for the race category.

Table 1: Final Scores

Category	# Labels	CNN3-Benchmark	CNN3 (with AE)	Best CNN*	Best CNN* (with Aug.)	Best CNN* (with PL)	Wav2Vec2-base
Accent	5	58.97%	57.43%	60.51%	59.71%	58.37%	**63.07%**
Age	5	51.79%	54.78%	82.73%	80.92%	**88.82%**	85.30%
Disease	2	69.07%	72.25%	91.21%	—	75.13%	**94.67%**
Emotion	6	46.43%	47.50%	54.43%	—	54.21%	**76.05%**
Race	3	72.16%	75.82%	89.75%	90.23%	89.28%	**94.63%**
Sex	2	92.21%	94.36%	97.85%	—	98.05%	**99.40%**

Looking at the 'Augmentation Method' column, where all methods come from the Librosa library, Time Shift, White Noise and Time Stretch were the only methods that produced results that were not significantly worse than the pre-augmentation score. On the other hand, other techniques experimented with include Pitch Scale and Random Gain which produced even worse results. Overall, it is clear that Augmentation did not yield notable results. As a result, we decided to use a pre-training method that does not involve manipulating our data called the pseudo-labeling.

The column 'Best CNN (with PL)' in Table 1 displays the results of pseudo-labeling using each category's respective best model. It is evident from the table that there was a slight decrease in F1 for Accent, Age, Disease, Race and Emotion tasks. The F1 for Age increased by almost 6% while the F1 for Sex only increased by less than 1%. The key point to emphasize with pseudo-labeling is how the quality, content and amount of unlabeled data must be carefully chosen to output the best model results.

Finally, the 'Wav2Vec2-base' column illustrates the F1 scores generated by Wav2Vec2. For the most part, this base transformer model delivered the best results. This can be attributed to the fact that there are almost 34 times more parameters in Wav2Vec2 than in a 9-layer CNN. However, on age data, the pre-trained model ResNet18 using pseudo-labeling performed best.

5.4 Graphs

The autoencoder training loss, which compares the input and output images, is depicted in the Figure 5. The training loss decreases as the number of epochs increases, while the output images become more and more similar to the input images.

In Figure 6 shows the GUI we designed. Users can simply input their voice through a microphone or upload an audio file. Then with their choice of categories and models, a Mel Spectrogram will be generated along with the predictions of each possible label. The CNN button will use the best CNN model from Table 1 to perform classification while the Transformer button will use the Wav2Vec 2.0 model. If the user wants to perform classification on different audio, they can simply 'Clear' the current file and upload another.

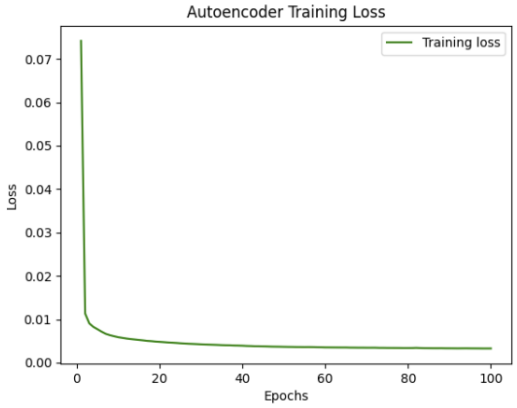

Figure 5: Autoencoder Training Loss

6 Discussion

While working on this project, we have run into many challenges concerning wrangling the data, pre-training as well model development. To begin with, the duration of each audio data for the accent task is too long to preprocess; thus, our approach was to split each audio file into separate, 5-second files(2). To make predictions, the scores of each 5-second segment will be averaged when testing on the original data for Parkinson's disease. For Accent, After making predictions on data that has been split, we used the most frequent prediction result (mode) as the final prediction result for each original audio file. If two prediction results have the same mode, we decided to use the prediction value with the label that has the higher number of original sample data. This method allowed us to maintain the integrity of the data whilst not influencing the accuracy of the classification. The same

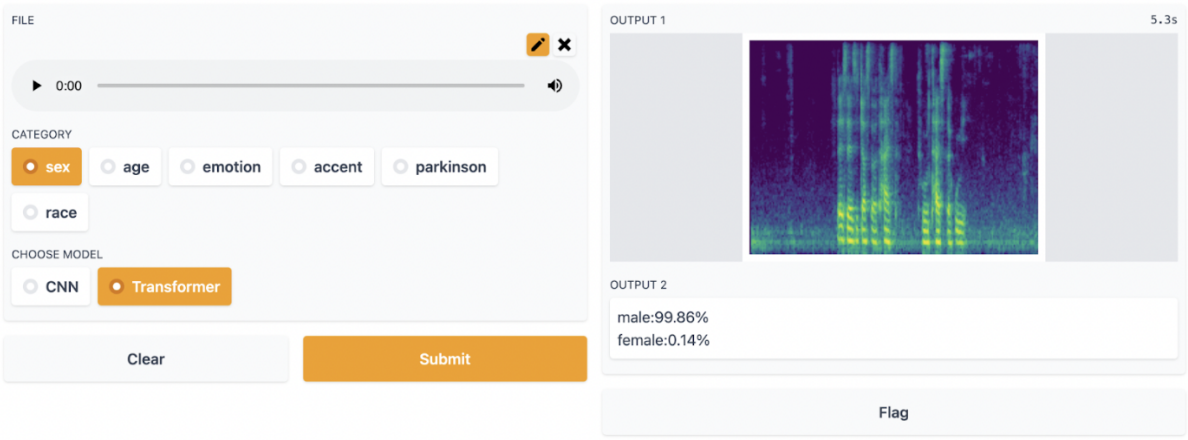

Figure 6: GUI Using Gradio Package

process was then implemented on the Parkinson's disease audio data after removing silent portions of the audio(1).

To address highly imbalanced datasets for the accent, race, and age categories, we have executed numerous augmentation methods to generate more samples of data. In order to improve the model accuracy, we have also generated Mel spectrograms for CNN models since it extracts additional features from audio and converts frequency into Mel Scale as opposed to regular spectrograms.

One of the project's constraints was that we did not have enough data for some categories, so we used the concept of pseudo-labeling in a slightly different approach than how it was done in related work. Although we had the same semi-supervised training logic, we used the same amount of unlabeled data as labeled data which is not ideal for relatively small datasets like ours - there should be significantly more unlabeled data inputted into the model for training. Upon continuation of this project, we plan to find additional data because we believe that this will facilitate training as the unlabeled loss can handle a higher alpha (weight) value.

On the same note, collecting unlabeled data to perform pseudo labeling was difficult as the quality and structure of the voice study should be similar to some degree to the labeled data. In the future, we plan to not only find significantly more data but also more relevant data to use as well as develop the training process using 1,000 epochs during pseudo-label training as this can benefit model performance exceptionally well.

Pre-training on the autoencoder was one of the core concepts that we explored in this project. The architecture we used was only a simple 3-layer model to determine whether accuracy could be affected. Since we found success with only 3 layers, we would like to increase the number of layers in the autoencoder to examine the influence of more complex architecture. On the other hand, when purely training directly on CNN models, we found that increasing the number of layers does not necessarily correlate with a higher score, so spending more time researching the intricacies of the autoencoder would be useful.

Out of all the models that we have tested, the transformer, Wav2Vec2-base is the best model for all categories. Future work will include experimenting with alternative transformers in the future such as UniSat, SEW, UniSpeech, UniSpeechSat, WavLM, Wav2Vec, Wav2Vec2Phoneme as these transformers have all been proven to perform exceptionally well with audio data.

7 Conclusion

The speech engine we have built is a fundamental step in developing speech recognition in industry. Audio data requires spending a significant amount of time in the development process on data preprocessing and pre-training. In our case, utilizing the autoencoder, augmentation and pseudo-labeling have yielded results that were not consistently positive. Using an autoencoder was incredibly useful as it gave us high success and demonstrated its usefulness with more research on a more complex architecture. Semi-supervised learning with pseudo-labeling was also rewarding for a single classification task with room for improvement by the addition of more data. Augmentation was not incredibly useful in this study particularly simply because audio data is very intricate to ma-

nipulate and see increasing results. Similarly, the use of Mel Spectrograms compared to regular spectrograms allowed us to recognize how computer vision is also a meticulous field which we can improve upon in future work by using a more sophisticated convolutional neural network for training.

In terms of downstream task results, classification of emotions was difficult to execute as predicting 6 labels on audio data where the only difference is very difficult to notice even by human standards. Using pseudo-labeling on the accent, race, emotion, sex and Parkinson's disease category to add more sample data, the predictions made by pre-trained and custom CNN models proved to be adequate regardless of the imbalanced class distribution. Although, age classification performed better with pseudo-labeling. On the other hand, the autoencoder benefited each task by a few percent in accuracy with the exception of the accent task. Unfortunately, all the augmentation methods we attempted to use did not produce better results on all tasks except emotions where the increase was less than 1%.

Ultimately, we successfully created an advanced speech engine to perform all the classification tasks and translated our work into a well-designed and performance-driven GUI that cohesively displays our results.

References

[1] Admin. Python remove silence in wav using librosa - librosa tutorial, Nov 2021.

[2] Alankar. Split audio on timestamps librosa, Feb 2020.

[3] Saad Albawi, Saad Al-Zawi, and Tareq Abed Mohammed. Understanding of a convolutional neural network, Aug 2017.

[4] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[5] Houwei Cao, David G. Cooper, Michael K. Keutmann, Ruben C. Gur, Ani Nenkova, and Ragini Verma. Crema-d: Crowd-sourced emotional multimodal actors dataset, Jul 2014.

[6] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[7] Arden Dertat. Applied deep learning - part 3: Autoencoders, Oct 2017.

[8] Dong hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks.

[9] Keith Ito and Linda Johnson. The lj speech dataset, 2017.

[10] Hagen Jaeger, Dhaval Trivedi, and Michael Stadtschnitzer. Mobile device voice recordings at king's college london (mdvr-kcl) from both early and advanced parkinson's disease patients and healthy controls, May 2019.

[11] Zhun Li, ByungSoo Ko, and Ho-Jin Choi. Naive semi-supervised deep learning using pseudo-label - peer-to-peer networking and applications, Dec 2018.

[12] Andrew Ng. Sparse autoencoder - stanford university, 2011.

[13] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, Dec 2015.

[14] pytorch.org. Conv2d. https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html.

[15] pytorch.org. Trans2d. https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html.

[16] Anirudh Shenoy. Pseudo-labeling to deal with small datasets-what, why amp; how?, Dec 2019.

[17] Rachael Tatman. Speech accent archive, Nov 2017.

[18] Velardo Valerio. Repository hosting code and slides of the audio data augmentation series on the sound of ai yt channel.

[19] Shengyun Wei, Shun Zou, Feifan Liao, and Weimin Lang. A comparison on data augmentation methods based on deep learning for audio classification.

[20] Wikipedia. Convolutional neural network, Dec 2015.

[21] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: An overview and application in radiology - insights into imaging, Jun 2018.