Analyzing the Performance of Existing Pre-Trained Audio Embeddings for Both Human Speech and Non-speech Acoustic Signals

A Major Qualifying Project (MQP) Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements
for the Degrees of Bachelor of Science in

Electrical and Computer Engineering
Computer Science

By:

Joshua Malcarne, Tyler Wong

Project Advisor:

Bashima Islam

Date: March 2024

Abstract

Embedded devices have limited resources but are required to perform multifaceted work, so it is crucial to minimize the number of resources needed for machine learning applications. In order to address this, we propose to create a versatile audio embedding that can perform both human and non-human speech tasks. We evaluate the performance of three potential pre-existing pre-trained audio embeddings: Wav2Vec, Wav2Vec2, and AST. Based on our results, Wav2Vec2 and Wav2Vec are suitable candidates. We further observe these models by fine-tuning on their non-specialized tasks. Through fine-tuning, we achieve 66.0% F1-score with Wav2Vec2 and 51.86% F1-Score with Wav2Vec for environmental sound classification.

Executive Summary

Machine learning has undoubtedly transformed the consumer technology industry and become ubiquitous in people's daily lives, especially when it comes to home-hub devices capable of human speech tasks, such as automatic speech recognition (ASR), speech diarisation, speaker recognition, and stutter detection. More than 64 million homes in the United States use smart home devices, equating to nearly 45% of all homes in the U.S. Of these devices, consumer electronics powered by voice-controlled virtual assistants are the most common devices in these homes [15]. To support the growth of the consumer technology industry and the demand for more services in these smart home devices, adding more features involving machine learning will aid in the predicted growth or even exceed it. However, adding more features will require storing more models for each specific task on the already resource-constrained devices. As home-hub devices have limited hardware resources, but are required to perform multifaceted works, it's crucial to minimize the number of resources required for machine learning applications. Therefore, there is a need to create a versatile audio embedding that can perform both human speech tasks and non-human speech tasks, simplifying the development process and minimizing memory usage.

When developing a single audio embedding for human and nonhuman speech classification tasks, we focus on three main objectives:

1. First, we determine pre-existing models that are likely to deliver positive outcomes for both classification tasks. We evaluate the performance of non-human acoustic audio tasks in environmental sound classification with Wav2Vec [3], Wav2Vec2 [3] and AST [4] using ESC-50 [9], Urbansound8k [7], AudioSet [10], and CHiME [21] datasets. We also evaluate the performance of human speech tasks using the LibriSpeech [12] dataset on the same models.

2. Next, to develop the best-performing model for both types of tasks, we fine-tune the optimal model using 3 different techniques to increase performance on non-specialized tasks while maintaining the performance of the specialized task. These techniques are direct fine-tuning on ESC-50, direct fine-tuning on AudioSet, and fine-tuning using Jialu et al.'s [16] pre-training methodology.

3. Finally, we briefly explore each model's performance on alternative human speech tasks such as speaker detection, speaker diarisation, child speech vocalization, and stutter detection. This is vital to exploring the potential of AST as a combined embedding for human speech and non-speech acoustic audio tasks. Due to time restrictions, this objective is not fully explored by our study, and will be explored further in future works.

	AST	Wav2Vec	Wav2Vec2
CHiME	93%	53%	40%
UrbanSound8k	89%	57%	21%
ESC-50	95%	35%	6.9%
AudioSet	11%	10%	1.0%

Table 1: F1-scores for classification inference on non-speech audio

	AST	Wav2Vec	Wav2Vec2
Librispeech (100hr)	73%	17.5%	5%

Table 2: WER for automated speech recognition (ASR) inference on human speech audio

While exploring our first objective, our observations are shown in tables 1 and 2. For reference, a higher F1-score indicates better performance in 1, and a lower WER indicates better performance in 2. Based on these results, we judge that Wav2Vec is the model most suitable for creating an embedding capable of both human speech and non-speech acoustic tasks. Wav2Vec's base model performs well on the ASR inference whilst achieving middle-of-the-line results on the classification of non-speech audio. While Wav2Vec is our

primary choice for further fine-tuning and observation, we also judge that Wav2Vec2 may be suitable for some fine-tuning. This is given Wav2Vec2's better performance on ASR inference and Wav2Vec2's performance on the classification of non-speech audio, which is worse but still within an acceptable margin.

After we determine that Wav2Vec is an optimal middle ground for performing speech and non-speech tasks, we move on to our second objective, fine-tuning the model for non-speech task optimization. We utilize fine-tuning techniques on the Wav2Vec audio embedding to improve performance for non-speech tasks, specifically ESC-50 and Audioset. We explore different non-dynamic and dynamic learning rates, the number of epochs, and the number of frozen layers for ESC-50 fine-tuning. For the AudioSet fine-tuning, we have explored different classifier architectures, along with techniques to prevent overfitting like regularizations and dropout layers. Ultimately, we achieve a 51.86% F1-score with fine-tuning Wav2Vec on ESC-50. Fine-tuning the Wav2Vec2 model on ESC-50 using Jialu et al.'s pre-training methodology [16] yields similar performance, achieving at best **66.0% accuracy** on the second layer, 62.0% accuracy on the output layer, and showing comparable results for the rest of the model. However, both fine-tuning results involve training on the full embedding, leading to greater degradation of human speech task effectiveness. Thus, we provide results of Wav2Vec's performance with different numbers of frozen layers, retaining some of the crucial weights for its specialized tasks.

Our team encountered several limiting factors throughout this study. One such limitation, limited access to sufficient compute resources, hindered our team while performing inference on AST. This was resolved by the project's Ph.D. student, who had access to resources on the WPI Turing Cluster. A more significant limitation we encountered is the suitability of the AST model when performing ASR inference. The AST model is designed to intake audio clips of fixed length, thus making it unsuitable for processing the Librispeech dataset [4]. We were unaware of this until later phases of our study, and while it does not invalidate our fine-tuning results, it does prompt further exploration into the potential of the AST model as a combined embedding for speech and non-speech tasks. Hence, our third objective of performing inference for alternate human speech tasks, such as speaker detection, speaker diarisation, child speech vocalization, and stutter detection. Our team also encountered limitations when using Jialu et al.'s methodology to train the Wav2vec model, as the encoder layers and feature aggregators of Wav2vec have different purposes and output different meaningful representations than Wav2Vec2, which can either be useful or useless for Jialu et al.'s method. Therefore, Jialu et al.'s practicality in future works with audio embedding other than Wav2Vec2 is limited for further exploration.

Our team encountered two additional limitations due to a lack of time in the later phases of our study. The first of these limitations was that we were unable to explore layer freezing for Wav2Vec2 using Jialu et al.'s pre-training methodology. Just as with the Wav2vec model, a study in this direction is important for maintaining Wav2Vec2's human speech task performance when fine-tuning for non-speech acoustic tasks using Jialu et al.'s algorithm. This is essential for creating an effective dual-embedding for both speech and non-speech tasks. In the future, we hope to add an algorithm to freeze some of the layers in the Wav2Vec2 model with Jialu et al.'s method. The other major limitation we encountered due to lack of time was that our team did not have the bandwidth to set up an adjusted pipeline for ASR inference on the fine-tuned embeddings. Because of this, we do not have results for how severe the degradation of human speech tasks is in the fine-tuned Wav2vec and Wav2Vec2 models. As a result, the observations borne of our results can not be established as success or not, without the evaluation of the speech-related tasks. In the future, we hope to have the performances of our fine-tuned models with speech-related tasks to determine the degradation of our models as well as their practicalities.

Acknowledgements

We would like to thank professor Bashima Islam and BASH LAB for their help, guidance, and assistance throughout the entirety of this project. Additionally we would like to thank Worcester Polytechnic Institute for providing us with the opportunity to complete this project.

Contents

1 Introduction **1**

1.1 Challenges . . . 2

1.2 Objectives . . . 3

1.3 Paper Structure . . . 4

2 Literature Review and Background Study **5**

2.1 Pre-trained Acoustic Models . . . 5

2.2 Acoustic Tasks with Pre-trained Model . . . 7

2.2.1 Speech Acoustic Tasks . . . 7

2.2.2 Non-speech Tasks . . . 8

2.2.3 Speech & Non-speech Tasks . . . 9

2.3 Datasets . . . 10

3 Methodology and Results **11**

3.1 Determining the Optimal Model Embedding . . . 12

3.1.1 Dataset Preparation and Model Description . . . 12

3.1.1.1 Wav2Vec2: UrbanSound8k . . . 12

3.1.1.2 Wav2Vec2: AudioSet . . . 13

3.1.1.3 Wav2Vec: UrbanSound8k . . . 14

3.1.1.4 Wav2Vec: AudioSet . . . 14

3.1.1.5 AST: UrbanSound8k . . . 15

3.1.1.6 AST: AudioSet . . . 15

3.1.1.7 Chime, ESC-50, and Librispeech . . . 16

3.1.2 Inference Results and Discussions . . . 16

3.2 Fine Tuning Wav2Vec and Wav2Vec2 . . . 17

3.2.1 ESC-50 for Wav2Vec Fine-tuning . . . 17

3.2.2 Fine-tuning Wav2Vec with ESC-50 . . . 17

3.2.3 Fine-tuning Wav2Vec with AudioSet . . . 19

3.2.4 Fine-tuning with Jialu et al.'s [16] Methodology . . . 20

3.2.5 Wav2Vec ESC-50 Fine-tuning Results and Discussions . . . 20

3.2.6 Wav2Vec AudioSet Fine-Tuning Results and Discussions . . . 24

3.2.7 Fine-tuning Results Utilizing Jialu et al.'s[16] Methodology . . . 26

3.3 Inference for Alternative Speech Tasks . . . 27

4 Limitations and Future Works **28**

4.1 Limited Compute Resources . . . 29

4.2 AST Model's Compatibility with ASR Task . . . 29

4.3 Jialu et al.'s[16] Algorithm Limitations on Wav2vec Architecture . . . 29

4.4 Layer Freezing for Wav2Vec2 Fine-tuning . . . 30

4.5 ASR Inference on Fine-tuned Models 30
5 Conclusion **30**
References **32**

List of Tables

1 F1-scores for classification inference on non-speech audio ii
2 WER for automated speech recognition (ASR) inference on human speech audio ii
3 Model size by number of parameters 5
4 F1-scores for classification inference on non-speech audio 16
5 WER for automated speech recognition (ASR) inference on human speech audio 16

List of Figures

1 Wav2Vec Performance on Different Learning Rates Using ESC-50 21
2 Wav2Vec Performance on Different Exponential Decay Rates Using ESC-50 22
3 Wav2Vec Performance on Different Linear Decay Rates Using ESC-50 23
4 Wav2Vec Performance on Different Number of Frozen Layers Using ESC-50 24
5 Wav2Vec Performance on Different Fine-Tuning Techniques Using AudioSet 25
6 Wav2Vec Performance on Different L2 Regularization Weight Decays Using AudioSet 26
7 Wav2Vec2 F1-score results after fine-tuning on ESC-50 27

1 Introduction

Machine learning is an increasingly popular subject in today's world of technology and has become unknowingly ubiquitous in people's daily lives. A Pegasystems Global study reveals that 77% of consumer electronics such as smartphones, smart speakers, smart TVs, wearable devices, and automobiles utilize some form of machine learning, compared to the 33% that consumers believe [13]. Machine learning has undoubtedly transformed the consumer technology industry, especially when it comes to home-hub devices capable of human speech tasks, such as automatic speech recognition (ASR), speech diarisation, speaker recognition, and stutter detection. According to a report from Research and Markets, the smart home market size is valued at 101.7 billion dollars in 2023 and is forecasted to reach 163.7 billion dollars by 2028 [14]. This growth can strongly be attributed to the implementation of applicable machine-learning (ML) models in these smart devices. In fact, more than 64 million homes in the United States use smart home devices, equating to nearly 45% of all homes in the U.S. In addition, the same blog shows that consumer electronics powered by voice-controlled virtual assistants are the most common devices in these homes [15]. Overall, this emphasizes the significant impact that human speech-based machine learning has on many people's lives.

Not only do ML algorithms created for human speech tasks revolutionize the consumer technology industry, but non-human speech models play a significant role. Some smart-home devices use these ML algorithms to trigger home automation services. For example, a door opening can trigger the smart home device to turn the lights on in a room [17]. Specifically, sound classification models can apply for safety and security, as statistics show that 26% of all smart home devices are being used for safety and security purposes, second to consumer technologies. This can involve the noises of windows shattering or doorbell ringing to ensure better the safety of a person's home[15]. Recent reports show that the global home security market is expected to grow from 29.67 billion dollars in 2023 to 78.92 billion dollars by 2023[18]. The smart home device's ability to classify these environmental sounds can be an attractive feature for a consumer and can strongly impact a person's daily tasks and way of living.

Machine learning's strong presence in these practical applications is due to their great strides in progress for better accuracy and performance, making them more reliable. In particular, a major achievement is Google's Word2Vec model embedding, a first of its kind in the machine learning industry. Word2Vec is a neural network that learns contextual information of words, which is represented as dense vector embeddings of words capable of capturing semantic and syntactic information [19]. This breakthrough has paved the way for more complex model embeddings that can perform recognition or classification tasks

for different acoustic tasks. However, these embeddings are developed with a single type of task in mind. To illustrate, breakthroughs like OpenAI's latest release of Whisper [1], and Facebook's Wav2Vec2 [2] are pushing the boundaries of accuracy and performance in audio tasks targeting human speech, e.g., automatic speech recognition (ASR), speech diarisation, child vocalization, speaker recognition, and stutter detection. Simultaneously, advancements in audio classification tasks, exemplified by models like AST [4], further refine our ability to categorize and understand environmental audio data.

To support the growth of the consumer technology industry and the demand for more services in these smart home devices, adding more features involving machine learning will aid in the predicted growth or even exceed it. However, adding more features will require storing more models for each specific task on the already resource-constrained devices. Though a study was conducted to create a system for environmental sound recognition for home automation, which involved a model for environmental sounds and an additional model for human speech tasks [17], it fails to consider resource-constrained smart home devices or other embedded systems. These devices cannot even support some popular and complex models, let alone multiple simple ones, since they are developed on expensive and powerful computers. To put it in perspective, the smallest version of audio to speech recognition (ASR) model Whisper [1] has 74M parameters and requires 139 MB of storage to store the mode. However, typical embedded systems made for home hub devices have 256KB to 4GB of storage, including the data, code, and model parameter memory, which is barely enough to hold a single deep learning model, let alone store and execute multiple models.

To support the multifaceted applications of these acoustic home-hub devices despite their limited resources, there is a need to create a versatile audio embedding that can perform both human speech tasks and non-human speech tasks, simplifying the development process and minimizing memory usage. Doing so will enable other productive features that enhance the device's value for consumers, making it more useful and ubiquitous. Furthermore, this encourages further research towards single model embedding specialized for multiple tasks beyond audio.

1.1 Challenges

Developing a single audio embedding for both human speech and non-speech acoustic tasks comes with challenges. In our project, we address three challenges:

1. Models for human speech and non-speech acoustic classifications are substantially different, as they have different structures. To illustrate, speech signals reside in a specific frequency zone and have more structures like phonemes, words, and phrases. On the other hand, non-human speech lacks such

structures and frequency ranges. For instance, Wav2Vec2, an audio embedding meant for speech-related tasks, uses contextualized representations following its feature extractor [2]. This part of the model allows it to understand how sounds are sequenced in language but is a structure less suitable for classifying non-human acoustic audio in the environment. Phonetics, the study of how humans produce and perceive speech sounds, is a crucial characteristic of speech that Wav2Vec2 uses. An example of phonetics is the word 'fish', which is made up of four letters but is sounded out to have three: 'f', 'i', and 'sh' sounds. Wav2Vec2's architecture enables the model to recognize these phonetic cues in speech, making it highly effective for human speech tasks, but leading to problematic phonetic features irrelevant for non-speech tasks.

2. Multiple datasets often use different fine-tuning techniques due to the distinct nuances of each dataset, which ultimately change a parameter from optimal to sub-optimal. It makes developing a fine-tuned model that is suitable for multiple datasets challenging and unpredictable. Additionally, fine-tuning models can be easily influenced by many factors, including the hyperparameters, size of datasets, and quality of the training data. In general, exact results are not reproducible but instead show a trend, which can make fine-tuning difficult, especially in knowing if a certain parameter is showing positive results or not.

3. The outputs of non-speech models are non-dynamic, which means the number of outputs of a specific task remains constant. In comparison, some human speech tasks, such as multiple speaker recognition and stutter detection tasks, are also non-dynamic, while other human speech tasks, such as ASR, are dynamic. For example, the output size of the multiple speaker recognition task we aim to explore later in this study has a constant output space, outputting 0 if there is one speaker or 1 if there are multiple speakers. Meanwhile, ASR applications require flexibility in their outputs. This eliminates the candidacy of any audio embedding specialized for ASR tasks, such as AST. We do not originally consider this challenge in our study, but in the later stages of our study, we begin to explore this.

1.2 Objectives

When developing a single audio embedding suitable for both human and nonhuman speech tasks, we focus on three main objectives:

1. Our first objective is to evaluate the performance of pre-existing models on both speech and non-speech tasks. To achieve this, we study three popular pre-trained models – Wav2Vec, Wav2Vec2 and AST – that are specifically developed for either of these types of tasks. We evaluate the performance of

non-human acoustic audio tasks in environmental sound classification using ESC-50, Urbansound8k, AudioSet, and CHiME datasets. We also evaluate the performance of human speech tasks using the LibriSpeech dataset on the same models. Our study shows that Wav2Vec is an optimal middle ground for performing speech and non-speech tasks.

2. Next, we fine-tune the Wav2Vec's pre-trained model on the ESC-50 dataset to develop a fine-tuned Wav2Vec version that performs well on non-speech tasks while maintaining the performance of the speech task. We then fine-tune this model using three different fine-tuning techniques to increase performance on non-specialized tasks while maintaining the performance of the specialized task. Our results show that we achieve 16.86% performance gain compared to the original embedding by fine-tuning Wav2Vec's hyperparameters on ESC-50. Fine-tuning the Wav2Vec2 model on ESC-50 using Jialu et al.'s pre-training methodology [16] yields similar performance, achieving at best 66.0% accuracy on the second layer, 62.0% accuracy on the output layer, and showing comparable results for the rest of the model. However, both fine-tuning results involve training on the full embedding, leading to greater degradation of human speech task effectiveness. Thus, we provide results of Wav2Vec's performance with different numbers of frozen layers, retaining some of the crucial weights for its specialized tasks.

3. Finally, we analyze the performance of alternative human speech tasks such as speaker detection, speaker diarisation, child speech vocalization, and stutter detection. This is vital to exploring the potential of AST as a combined embedding for human speech and non-speech acoustic audio tasks. Due to time restrictions, this objective is not fully explored by our study, and will be explored further in future works.

1.3 Paper Structure

Section 2 outlines the background and related works of the proposed work, including the models and datasets we study throughout this project. Section 3 elaborates our methodology in two parts, one for each objective explained in section 1.2. We provide a detailed explanation of our process of achieving these objectives and present an in-depth explanation of the results we produce. Section 4 discusses our observations and limitations based on the results along with the future plans for the project. Finally, Section 5 concludes this paper with our brief summary of our findings and final thoughts.

2 Literature Review and Background Study

This section describes existing work on developing and characterizing efficient networks for multi-faceted audio classification tasks. We begin by discussing the current state-of-the-art pre-trained acoustic models we explore in section 2.1. Next, we present the several acoustic tasks we perform with these pre-trained models in section 2.2, encompassing human speech tasks in section 2.2.1, non-speech acoustic tasks in section 2.2.2, and the combination of speech and non-speech tasks introduced by the Whisper-AT model in section 2.2.3. Finally, we introduce the datasets that we explore for inference and fine-tuning on these audio tasks in section 2.3.

2.1 Pre-trained Acoustic Models

	AST	Wav2Vec	Wav2Vec2	Whisper
Model Size	87 M	40 M	317 M	74 M
Memory Size	336 MB	152 MB MB	900 MB	282 MB

Table 3: Model size by number of parameters

Wav2Vec [3]. Wav2Vec is one of the unsupervised acoustic embeddings developed by Facebook for speech recognition to address the common challenge of having a limited training dataset. Wav2Vec consists of an encoder network that would transform audio signals into latent representations using a five-layer convolutional network. Following the encoder, a context network contextualizes the representations using nine fully connected layers. Since Wav2Vec relies on unsupervised pre-training, the model incorporates affine transformations and a contrastive loss function to distinguish between a positive and negative example. This approach is the catalyst for efficiently using unlabeled data, providing valuable representations for downstream tasks in speech recognition[3].

Wav2Vec2 [2]. Wav2Vec2 is the improved model of the Wav2vec specialized for human speech tasks that exploit the phonemes of human speech. Wav2Vec2 has a multi-layered convolutional neural network responsible for encoding speech audio, followed by a Transformer network that learns the contextual representations, capturing the dependencies of an entire speech data. Next, the model uses the Gumbel softmax to represent the latent speech features during the self-supervised learning process and shows promising results compared to nonquantized units. The training process involves masking the latent feature representations and employing them in a contrastive task that distinguishes between the significant and insignificant latent representations. The contrastive task relies on a codebook to pick out the positive and negative examples, and to ensure codebook entries are used uniformly, diversity loss is utilized [2]. Wav2Vec2 revolutionized

speech recognition technology, allowing more possibilities for the model to excel in a range of tasks, including language translations, dialect recognition, and even the classification of emotions.

AST [4]. Audio Spectrogram Transformer (AST) is an audio classification model that provides an embedding leveraging a simple attention-based architecture instead of a conventional convolutional neural network (CNN). It converts the input audio waveform into a sequence of Mel filterbank features, creating a spectrogram, which is then split into a sequence of patches that are flattened using a linear projection layer. The sequence is then augmented with a [CLS] token, a special type of classification token used in tasks such as text and sequence classification. After this, the augmented sequence is fed into a Transformer encoder. The Transformer's output serves as the audio spectrogram representation and is subsequently used for classification via a linear layer with sigmoid activation. AST's simple architecture requires fewer parameters than convolution-based audio classification models, facilitating quicker convergence during training. AST is a groundbreaking approach to audio classification and maintains state-of-the-art results on benchmarks such as AudioSet, ESC-50, and Speech Commands V2 despite its unorthodox architecture [4].

Whisper [1]. Whisper is a model embedding designed by OpenAI for diverse human speech tasks, including the commonly encountered automated speech recognition (ASR) tasks. The embedding is trained on 680,000 hours of real-world speech data from different tasks and languages. The model is based on common sequence-to-sequence architecture featuring a Transformer-based encoder and decoder. The encoder processes input audio features into a fixed-size context vector, which the decoder takes as input to generate the corresponding text transcription. Whisper's architecture supports multitask learning, allowing the model to simultaneously learn multiple tasks or languages by training on a diverse range of datasets. To this effect, the model is pre-trained on such a collection of large and diverse datasets, conferring robustness in the model's capabilities. To handle these differing datasets, strategies such as text normalization are employed to improve recognition accuracy, and reliable long-form transcription techniques such as beam search, temperature adjustment, and voice activity detection are implemented to improve transcription accuracy by the model. The complex set-up of the Whisper architecture, focused on multitasking and multilingual learning, allows the model inherent flexibility for human speech tasks, enabling the architecture for speech tasks separate from common ASR, which is further detailed in Section 2.1. While we will not perform inference on the Whisper model in this study, the model's comprehensive approach to speech recognition, leveraging scaling, multitask learning, text normalization, and decoding heuristics to achieve robust performance across various speech tasks and languages are insightful to the goal of creating a single robust embedding capable of both human speech and non-speech acoustic tasks.

2.2 Acoustic Tasks with Pre-trained Model

As mentioned in Section 1, the first objective of our study focuses on determining the capabilities of pre-trained acoustic models for different acoustic tasks. This includes human speech tasks, non-speech acoustic tasks, and the capability of a model to perform both. This subsection will introduce instances and applications of these audio tasks relevant to our study in-depth, starting with speech tasks in section 2.2.1, followed by non-speech tasks in section 2.2.2, before finally concluding with a discussion of the convergence of human speech and non-speech acoustic tasks observed in the whisper-AT model in section 2.2.3.

2.2.1 Speech Acoustic Tasks

Automated Speech Recognition. Automated speech recognition (ASR) is a common human speech task focused on converting spoken language directly into written text. This task has applications in voice-controlled devices, speech-to-text programs, customer service automation, devices for human accessibility, and more. To find a real-world implementation of ASR, look no further than Apple's Siri, Google Voice, or Microsoft Cortana. Models with ASR capabilities are already widespread in day-to-day life, and hardware- and software-based noise reduction techniques make ASR in noisier real-world settings possible. The dataset we explore throughout this study, LibriSpeech, is an online corpus of paired speech and corresponding text sourced from online, publicly available audiobooks. The outputs of ASR models are often dynamic since the output of speech-to-text changes length based on the length of the spoken segment [12].

Speaker Recognition. Speaker recognition, also called speaker identification, is a human speech task aimed at identifying whether a speaker in an audio slice is a certain individual or not. This task has applications in forensics, security, voice authentication, searches for individuals via large collections of audio clips, and more. Oftentimes, these applications are restricted by noisy and uncontrolled environments of audio slices used, compounding the challenge of making speaker recognition viable in real-world situations. The dataset we explore in the later stages of this study, Vox Celeb1, is a speaker recognition dataset with many audio clips from over a thousand individuals compiled from YouTube. The outputs of speaker recognition models are often non-dynamic since the output of the network is often a yes-no classification of whether the speaker's identity matches the target [11].

Stutter Detection. Stutter detection is a human speech task that focuses on the detection of speech disfluency in audio slices. It is an area of research very important to the accessibility of speech-controlled technology, as such technology is and continues to become more prominent in everyday life on both personal and public devices. The largest bottleneck to the current research and development of stutter detection tasks

is the lack of labeled data in circulation. Stutter data is challenging to label due to the diversity of speech disfluency manifestations in individuals, and data for speech disfluency is often kept private [6]. While we do not explore stutter detection tasks directly in this study, future work on this project will observe the inference scores of the Wav2vec2, Wav2vec, and AST models for stutter detection tasks. These observations will help to inform decisions on the viability of AST for human speech tasks of non-dynamic output.

Speaker Diarisation. Speaker diarisation involves tagging of audio interactions between family members in a home environment. This task has applications that are prevalent in observing infant mental health and development, monitoring relationships between family members, and infant cry detection, among other things. Audio for speaker diarisation tasks is often recorded in household environments from local families. Jialu et al. [**?**, **?**] then explore utilizing this mass of collected data to perform large-scale unsupervised pretraining on the Wav2Vec2 model by first extracting the features of the unlabeled audio data by passing it to the Wav2Vec2 embeddings. Finally, Jialu et al. train the embeddings based on the average cross-entropy of 3 classifiers: one for each of mother-infant, father-infant, and sibling-infant interactions. While we do not explore speaker diarisation tasks in this study, this specific unsupervised pre-training approach for Wav2Vec2 is of great interest. We use this pre-training approach on Wav2Vec2 with non-speech data from the ESC-50 data set in order to observe whether it confers robustness to the Wav2Vec2 embedding for achieving both human speech and non-speech acoustic tasks. Because model outputs can vary, whether speaker diarisation tasks are dynamic or non-dynamic depends on the specific task a model is built to achieve.

2.2.2 Non-speech Tasks

Environmental Sound Classification. Environmental sound classification is a common non-speech acoustic task that pertains to classifying sounds to specific pre-determined classes. This flexible acoustic task has a variety of real-world applications, ranging from contextualizing noise environments to enhancing automated surveillance systems with acoustic sensors [20]. Datasets designed for environmental sound classification are a main focus of our study. UrbanSound8k is a dataset focusing on ten sound labels commonly encountered in urban environments, including car horns, sirens, and jackhammers [7]. The ESC-50 dataset similarly has five different sets of ten sound labels, encompassing animals, natural soundscapes and water sounds, non-speech human sounds, interior domestic sounds, and exterior urban noises [9]. CHiME is a series of challenges and corresponding datasets released with the goal of advancing speech and non-speech task technology. CHiME-5 track one, which we use in our paper, contains recordings of twenty dinner parties held in different rooms, sporting six labels for child speech, adult speech, TV, percussion, broad, and silence [21] Google's AudioSet is a massive dataset sourced from YouTube videos, which includes 527 diverse audio labels from a more ro-

bust environmental sound classification network [10]. We utilize these datasets for inference and fine-tuning purposes throughout the majority of this study, and we discuss the individual datasets in more depth in section 2.3. Because the labels for environmental sound classification tasks are often pre-determined, the output space of a corresponding model is constant, making this task non-dynamic.

Household Sound Classification. Household sound classification is a form of environmental sound classification focusing on household environments. Applications of household sound classification are similarly focused on in-household applications, such as acoustic sensors for automatic security systems [20], environment monitoring for infant sounds on baby monitors, and conversational speech recognition and contextualization [21]. Household sound classification datasets we explore in this study include ESC-50, which includes a set of ten labels for interior domestic sounds [9], and CHiME-5 track one, which aims to contextualize the source of speech within different rooms of a household [21]. Google AudioSet also contains labels that can be used for household sound classification, although the dataset also contains labels outside the scope of this more specified task [10]. Just as with environmental sound classification, household sound classification tasks often use labels that are pre-determined, and the output space of corresponding models is thus constant, making this task non-dynamic.

2.2.3 Speech & Non-speech Tasks

Whisper-AT [5]. Whisper-AT is a model created by modifying Whisper and is one of the first works that explores non-speech task performance on a speech embedding model. Whisper, a model embedding we discuss in section 2.1, is trained using a diverse dataset consisting of 680,000 hours of real-world speech data, which confers robustness in the face of various background noises commonly encountered in real-life scenarios. Upon scrutinizing the Whisper model, researchers observe that its audio representation, while not entirely noise invariant, correlates significantly with non-speech sounds. With this discovery, they are able to conclude that the Whisper model is an exceptional backbone for not only speech recognition tasks but audio classification tasks as well. By preserving Whisper's parameters by freezing the layers, they retain the model's correlation to non-speech sounds. They connect it to a supplementary network that includes meaning pooling, linear projection, and a temporal Transformer at each Whisper layer, which is further classified using a multilayer perceptron. This new model, Whisper-AT, is capable of both the human speech tasks of the original model and non-speech acoustic tasks. Whisper-AT produces competitive results for audio classification tasks, particularly in terms of efficiency, compared to models catered for audio classification, showing that it is possible for a single acoustic model to perform well on both types of tasks. However, while Whisper-AT achieves state-of-the-art results for ASR tasks, its performance on other human speech tasks,

such as stutter detection and speaker detection, are unknown. These alternative tasks are important for assessing the capability of Whisper-AT as a dual-purpose model.

2.3 Datasets

ESC-50 [9]. The ESC-50 dataset is a widely used collection of audio recordings that are extracted from public field recordings by the FreeSound.org project. This dataset consists of 2000 audio clips formatted in WAV files, each of five seconds in duration, labeled for environmental sound classifications. ESC-50 comprises fifty different classes that are organized into five primary ones: animals, soundscapes and water sounds, human noises, interior/domestic sounds, and urban sounds. ESC-50 has five folds, where every data sample is ensured to be in one fold only. However, for the purposes of this project, we combine all ESC-50 folds into one and perform a train-test-validation split instead of a 5-fold-cross-validation.

AudioSet [10]. Google's AudioSet is a multilabel dataset for acoustic event recognition research. The collection consists of over 2 million 10-second audio recording clips sourced from YouTube videos, with each clip labeled as one or multiple of 527 possible sound event classes. This includes both speech and non-speech audio data, and the distribution of labels across the entire dataset is not balanced. AudioSet is compiled by the Sound Understanding group in the Machine Perception Research organization at Google. We utilize a more balanced distribution of labels across 19,645 of the compiled audio clips and process them using a sampling rate of 1600 Hz over the 10 seconds in each clip. For both finetuning and inference, we populate our models via cross-validation over five folds.

CHiME [21]. CHiME is a series of challenges aimed at further developing speech and non-speech task technology and provides datasets for contestants to use to compete with each other. We specifically utilize CHiME-5 track one, containing twenty dinner parties' recordings. Each dinner party has four people and must take place in the kitchen, dining room, and living room for at least two hours. The dataset has 50 hours of audio clips that can be used for various tasks, but we use six different labels classifying a child, adult, TV, percussion, broad, and silence. Similar to ESC-50, we perform a train-test-validation split for our tests.

UrbanSound8k [7]. UrbanSound8k is a dataset consisting of 8732 labeled four-second-or-less audio clips of urban sounds from 10 different classes formatted in WAV files. As provided by the UrbanSound8k website, the dataset is organized into ten folders, each containing a random amount of audio clips from different classes. The website recommends using 10-fold cross-validation and advises against reshuffling the data. We perform the 10-fold cross-validation to these predefined splits and average the evaluation metrics so we do

not inflate scores that don't represent the model's performance. This approach involves training the model on nine folds and testing the remaining fold, repeating the process ten times using different folds as tests in each iteration.

Librispeech ASR [12]. LibriSpeech ASR is a widely used dataset for training and evaluating automatic speech recognition (ASR) systems, a common human speech task. The dataset comprises approximately 1000 hours of English speech data sourced from publicly available audiobooks and is segmented into categories comprising clean speech and speech with various levels of simulated noise. LibriSpeech was compiled by a team at the Center for Language and Speech Processing & Human Language Technology Center of Excellence at The John Hopkins University. We utilize an 80:20 train-test split when performing inference on our models using the LibriSpeech dataset.

Vox Celeb1 [11]. Vox Celeb1 is a dataset tailored for speaker recognition research. It consists of over 100,000 total speech clips extracted from YouTube videos, with multiple clips collected for each unique speaker. The dataset is compiled by the Visual Geometry Group from the Department of Engineering Science at the University of Oxford, UK, and is presented in INTERSPEECH 2017. For each of the dataset's 1251 speakers, we sample 12 audio clips, matching six together for each speaker and pairing the other six off randomly. This results in a balanced split between same-speaker and different-speaker data in our inference pipeline. When performing inference with this dataset, we use Equal Error Rate (EER) as our success metric rather than F1-score, unlike all other datasets discussed thus far.

3 Methodology and Results

This section discusses our study's methodology in depth. We first consider determining the optimal embedding for creating a combined embedding for speech and non-speech data in Section 3.1 using the Wav2vec, Wav2Vec2, and AST models. Next, we provide an analysis of our inference results and subsequent decisions, moving into the fine-tuning section of our study. Section 3.2 describes the three fine-tuning approaches on Wav2vec and Wav2vec2 models and provides insights on the results of fine-tuning and observations regarding hyperparameters and the overall success of these models. Finally, we introduce the beginning of this study's efforts to gauge inference on all three models' performance for several human speech tasks, e.g., stutter detection and speaker recognition in Section 3.3.

3.1 Determining the Optimal Model Embedding

We investigate three potential pre-existing model embeddings for fine-tuning, aiming to create a model embedding capable of doing both ambient sound classification and speech recognition tasks. The three models under consideration are AST, Wav2Vec2, and Wav2vec. We evaluate these models based on their strengths in specific domains: AST excels in audio classifications, while Wav2Vec and WavVec2 are designed for speech recognition tasks. We are testing four different audio classification datasets: UrbanSound8K, ESC-50, AudioSet, and Chime. Simultaneously, we are utilizing the LibriSpeech dataset for speech recognition as well. Since we anticipate that Wav2Vec will serve as the optimal middle ground, the choices of our dataset reflect our emphasis on audio classification, as Wav2Vec is tailored explicitly toward speech recognition tasks. The following outlines the procedures we employ to evaluate these models using the selected datasets and the challenges we encounter.

3.1.1 Dataset Preparation and Model Description

3.1.1.1 Wav2Vec2: UrbanSound8k

Data Preparation. To evaluate Wav2Vec2 on UrbanSound8K, we first prepare the data by developing a function to format the UrbanSound8K dataset while preserving its 10-fold structure. The initial step involves retrieving the WAV files one fold at a time and converting the WAV files of audio clips into floating-point time series arrays sampled at 16 kHz. For each fold, we store the resulting audio arrays in a data frame and encode their associated labels using one-hot encoding. The data frames are then stored in an array that will save the 10 data frames associated with each fold in the dataset.

Model Architecture. Subsequently, we construct a classification model for Wav2Vec2 that will be placed after the Wav2Vec2 pre-trained embedding. This model consists of Wav2Vec2's feature extractor connected to a fully connected network, incorporating a dropout layer with a 0.1 drop rate and a dense layer consistent with 768 nodes to the number of labels (10). A sigmoid function is applied to the final results to facilitate feature classification.

Training Method. The data frames are organized into nine for training and one for testing. We iteratively load the data into the model, changing the test data frame each iteration for the tenfold cross-validation. The audio arrays are then resampled to 16KHz again to ensure that all audio arrays are the same size. Since the audio arrays are the same length, they are then vertically stacked and are fed through a DataLoader function provided by Pytorch that prepares data for the model. The pre-trained feature extractor of Wav2Vec2 is

loaded in from HuggingFace and is connected to the classifier model we created. The chosen hyperparameters include a learning rate of 5e-5 with an AdamW optimizer, batch size 32, three epochs, and binary cross-entropy loss on logits, given that the audio clips are not multi-labeled. We only train the classifier model in the training process and freeze the Wav2Vec2 feature extractors. Once the model is trained, we evaluate it using the average F1-score for all ten folds used as tests.

Challenges. One significant challenge we encounter is the inconsistencies in the duration of the audio recordings in UrbanSound8K, which causes variations in the length of the audio arrays. These variations in length prevent the audio array from being able to be placed in a 2D array/tensor, which is a requirement for an input of any machine learning model. To address this, we resample all audio arrays to 16 kHz to fulfill this requirement. In this experiment, any loss of information or temporal distortions resulting from the resampling of audio arrays is disregarded.

3.1.1.2 Wav2Vec2: AudioSet

Data Preparation. To perform inference with the Wav2Vec2 model on the AudioSet dataset, we first prepare the data. To do so, we utilize two csv files (class_labels_indices.csv and train.csv) containing the class labels and .wav file names consecutively. We first create a ground-truth onehot encoding for each of the .wav files before then preprocessing all ten second .wav files to a sampling rate of 16000 Hz using the Python library librosa. During this process, we encounter an error caused by the existence of audio files of non-standard length. To fix this, files of non-equivalent length are dropped from the dataset and csv's. After this, we define a custom dataset class inheriting from torch.Dataset, and split our preprocessed audio data into five folds using sklearn.model_selection.KFold.

Model Architecture. After preparing the data, we load the model and define a corresponding classifier for the output embeddings. This classifier is tacked onto the end of our model and consists of a dropout layer with a dropout rate of 10% followed by two dense linear layers.

Training Method. For the train and test data in the folds, we populate the model with the data and feed the output embedding into the classifier. Due to memory restrictions, this is done with a batch size of eight for three epochs. Following this, we train the classifier by comparing its output to the earlier-defined ground truth one hot encoding. We encounter issues with the initial higher learning rate of 5e-4, which results in NAN weights and an F1-score of 0. This is most likely due to the exploding gradient problem. To fix this, we tune the learning rate to 5e-12, fixing the issue. The optimizer we use for training the classifier is Adam, and the embedded model itself is not modified during this process. Finally, we observe the F1-score for the

model by classifying the output embedding for the testing data and comparing it to the ground truth one hot encoding.

3.1.1.3 Wav2Vec: UrbanSound8k

Data Preparation. To evaluate Wav2Vec, we follow a similar process to what we perform with Wav2Vec2. We prepare the data exactly in Section 3.1.1.1, where we process the UrbanSound8K data. The preprocessed data passes through the model embeddings of Wav2Vec.

Model Architecture. We download the weights of the feature extractor of Wav2Vec from a GitHub source [6]. After the feature extractor, we develop a classification model comprising a dense layer with 50,176 parameters and 500 neurons. We apply the ReLU function to the output of this dense layer, followed by a dropout layer of 0.2. Then, we add another dense layer with 500 neurons and 10 outputs and apply the softmax function to the results. We resample the data to 16KHz, the same way we did for Wav2Vec2, and use the Dataloader function to prepare the data for Wav2Vec.

Training Method. The chosen hyperparameters are a batch size of 32, epochs of 5, learning rate of 1e-4, and use the Adam optimizer. The model is trained by putting the data through the Wav2Vec feature extractor and into our classifier model. The loss is calculated, and backpropagation is only done to the classifier model, as this training is only for the classifier we created. Unlike Wav2Vec2, we decide not to iterative do the K-Fold and run nine of the datasets for training and one fold for the validation. This is to make the process of evaluating Wav2vec performance faster rather than iterating through all the possible combinations.

Challenges. There is a minor challenge where the feature extractor results do not match the tensor shape of our classification model, which is resolved by printing the shape of the tensor after the feature extractor and copying the shape of the tensor as the size for the classifier model.

3.1.1.4 Wav2Vec: AudioSet

Data Preparation. To perform inference with the Wav2Vec model on the AudioSet dataset, we prepare the data using a modified version of the data pipeline discussed in Section 3.1.1.2. We define a custom dataset class inheriting from torch.Dataset. For the Wav2Vec model, we do not allocate the data into folds, and instead, we only split the data 80% train and 20% test.

Model Architecture. After preparing the data, we load the model and define a corresponding classifier for the output embeddings. This classifier is tacked onto the end of our model and consists of a dropout

layer with a dropout rate of 5% followed by a ReLU layer, a Sigmoid layer, and 2 dense linear layers.

Training Method. We populate the model with our train and test data and feed the output embeddings to the classifier. This is performed with a batch size of eight for five epochs. Following this, we train the classifier by comparing its output to the earlier defined ground truth one hot encoding, using a tuned learning rate of 1e-4. The optimizer we use for training the classifier is Adam, and the embedded model itself is not modified during this process. Finally, we observe the F1-score for the model by printing out a sklearn.metrics classification report comparing the output from the classifier on test data to the ground truth onehot encoding.

3.1.1.5 AST: UrbanSound8k

Data Preparation. When performing analysis for AST with the UrbanSound8K dataset, we utilize a GitHub repository that contains prewritten bash scripts [4]. These scripts are designed to work as pipelines for the AST model to assess its performance on different datasets. However, the input datasets must be in a specific format for the scripts to work correctly. The bash scripts conveniently use K-fold validation, which is already formatted in the UrbanSound8K dataset. Since there are ten different folds in UrbanSound8k, we have to create twenty different JSON files for the ten-fold validation. A total of twenty JSON files are used for the dataset. Out of these, ten files are used for evaluation data, each containing one of the ten folds of the dataset. The remaining ten JSON files are used for training data associated with the evaluation JSON. The training data will exclude the fold that is being used as evaluation data. Each JSON file contains a dictionary with a 'data' field.' In the 'data' field, there is an array of dictionaries, each of which contains the directory of the WAV file and its label.

Challenges. These labels are formatted as "/m/" followed by an assigned integer to the label. We can not use Bash commands on Windows, so we have to switch to Ubuntu, an open-source Linux operating system, to run the scripts provided by the GitHub repository. However, running these scripts for AST requires more GPU memory, which prevents us from conducting the test on our own local machines. To solve this issue, we run the scripts on a more powerful computer, WPI's Turing Cluster.

3.1.1.6 AST: AudioSet

Data Preparation. To perform inference with the AST model on the AudioSet dataset, we prepare the data in accordance with a pre-existing pipeline requiring JSON input in 10 folds. This is the same K-fold validation method we discuss in Section 3.1.1.5.

3.1.1.7 Chime, ESC-50, and Librispeech

Data Preparation. The pipelines for each of these tasks are similar to the pipelines we describe throughout Section 3.1 [1]. For example, because Chime and AudioSet are both multilabel datasets, the data preparation and inference pipelines for each of the three models are similar for these datasets (refer to Sections 3.1.1.2, 3.1.1.4, and 3.1.1.6). This is also the case for ESC-50 and UrbanSound8k, both of which are single-label datasets (refer to Sections 3.1.1.1, 3.1.1.3, and 3.1.1.5). The results inference results we obtain on Wav2Vec2, Wav2Vec, and AST for the Chime and ESC-50 datasets, paired with our results for the AudioSet and UrbanSound8k datasets, are important in helping us determine the optimal embedding for further fine-tuning and development in this study.

Inference is also performed on the Librispeech dataset in order to observe each model's effectiveness on human speech audio [2]. These ASR pipelines are not comparable to the other pipelines discussed throughout Section 3.1 and use Word Error Rate (WER) as their metric of success. When obtaining results with this pipeline, lower WER is optimal.

3.1.2 Inference Results and Discussions

	AST	Wav2Vec	Wav2Vec2
CHiME	93%	53%	40%
UrbanSound8k	89%	57%	21%
ESC-50	95%	35%	6.9%
AudioSet	11%	10%	1.0%

Table 4: F1-scores for classification inference on non-speech audio

	AST	Wav2Vec	Wav2Vec2
Librispeech (100hr)	73%	17.5%	5%

Table 5: WER for automated speech recognition (ASR) inference on human speech audio

Performing inference on the models using the pipelines discussed throughout Section 3.1 yields the results seen in tables 4 and 5. For non-speech audio inference, we observe that AST performs the best on all datasets. Wav2Vec2 performs the worst out of the models, whilst Wav2Vec achieves scores between AST and Wav2Vec2. For speech audio inference, the results are flipped, with Wav2Vec2 performing the best, Wav2Vec performing in between the other models, and AST performing the worst by far.

All models perform poorly on AudioSet, with AST achieving the highest accuracy at 11%. This

[1] Inference on Wav2Vec2, Wav2Vec, and AST for the Chime and ESC-50 datasets is handled by a Ph.D. student working on the project.

[2] Inference on the Librispeech dataset is handled by the same Ph.D. student as the Chime and ESC-50 datasets.

is likely due to the number of class labels in AudioSet, totaling 527. In comparison, UrbanSound8k and ESC-50, which achieve much better classification results during inference, each have a total of ten class labels. Chime similarly possesses seven class labels and achieves much better classification results during inference than AudioSet despite both datasets being multilabel.

Based on these results, we judge that Wav2Vec is the model most suitable for creating an embedding capable of both human speech and non-speech acoustic tasks. Wav2Vec's base model performs well on the ASR inference whilst achieving middle-of-the-line results on the classification of non-speech audio. While Wav2Vec is our primary choice for further fine-tuning and observation, we also judge that Wav2Vec2 may be suitable for some fine-tuning. This is given Wav2Vec2's better performance on ASR inference, and Wav2Vec2's performance on classification of non-speech audio, which is worse but still within an acceptable margin.

3.2 Fine Tuning Wav2Vec and Wav2Vec2

After experimenting with Wav2Vec, Wav2Vec2, and AST, our results show that Wav2Vec is the best candidate for good middle ground between human speech and non-speech acoustic tasks. Wav2Vec2 also shows some potential for being another candidate, as previous studies have displayed that it can be robust against noise, which was proven beneficial for non-human speech classifications [6]. This section breaks down the different fine-tuning techniques we used for Wav2Vec and Wav2vec2.

3.2.1 ESC-50 for Wav2Vec Fine-tuning

In our efforts to make the process of fine-tuning Wav2Vec simpler, we decide to use ESC-50 as our dataset for fine-tuning. ESC-50 contains 2000 data samples, which is sufficient to adjust the weights of Wav2Vec. Furthermore, it is small enough to enable us to run multiple tests without taking too much time to train the model. Once we complete fine-tuning with ESC-50, we plan to move to Audioset because it is a significantly larger dataset with over 100,000 data samples. This enables us to observe significant changes to the weights of Wav2vec and assess the impact of its performance on the ESC-50 dataset.

3.2.2 Fine-tuning Wav2Vec with ESC-50

The first hyperparameter we explore is the number of epochs used for training by doubling the original 30 to 60 epochs to test the performance. The number of epochs is the easiest hyperparameter to increase the performance of a model and ensure that the model has enough training time to learn the pattern

of ESC-50. However, we keep in mind that increasing the number of epochs too much can lead to overfitting, so we pay close attention to the training loss and validation loss.

Another fine-tuning technique we investigate is the learning rate, which is set initially as 1e-4 on every Wav2Vec evaluation we performed. We theorize that finding the optimal learning rate will promote a decrease in training and validation loss, thus increasing the model's accuracy. We start by decreasing the learning rate with the denominations of 0.5 starting from 1e-4 with the number of Wav2Vec layers frozen as 27 and 25 layers. At the same time, the rest of the parameters and classification model will remain the same as Wav2Vec evaluations in Section **??**. The frozen layers are added to investigate if learning rates and frozen layers have an impact on each other. Frozen layers will be further explained in the next paragraph. We also look into dynamic learning rates, which are learning rates that will decrease by a factor every certain number of epochs during the training process. Dynamic learning rates are often used to avoid overshooting the minimum loss function, allowing the learning rate to adapt to the training over time. Specifically, we experiment with linear step decay and exponential decay. For the linear step decay, we observe the performance impacts of different times the step decay will occur and the amount of decay it will do to the learning. First, we test different amounts of epochs. Step decay with a denomination of 5 epochs occurs while keeping the decay rate at 0.5, so we can find the best-performing number of epochs for step decay. Then, we keep the best performing epoch number as a constant and vary the decay rate with 0.1, 0.5, and 0.9, which are the most used step decay rates. For exponential decay, we experiment with 0.5, 0.8, 0.9, and 0.95. We compare the accuracy of the two dynamic learning rate strategies to determine the most optimal strategy.

Next, the optimal learning rate and epochs are used with layer freezing. Layer freezing involves freezing the weights of particular layers, usually the initial ones, during model training. The purpose of this technique is to retain the characteristics of the weights that help support the performance of its specialized task. We want to maintain some of Wav2Vec's performance with human speech tasks by freezing the starting layers. Meanwhile, we aim to improve the performance of non-human speech classification by training the remaining layer weights of Wav2Vec and the classification model. We want to identify the optimal numbers of layers frozen in Wav2Vec that result in the highest accuracies by experimenting with different numbers of layers frozen. Since Wav2Vec has 30 total layers in the audio embedding, we freeze layers incrementally from 0 to 30 in step 5, starting from the embedding's first layer. We will observe each one's accuracy. We plan to test all frozen layer model embedding with the ASR task and with the Librispeech dataset; however, due to time constraints, we are unable to achieve this step.

3.2.3 Fine-tuning Wav2Vec with AudioSet

For fine-tuning with AudioSet, we initially start with the most optimal fine-tuning parameters, which include the dynamic learning rate and number of epochs we discover during the fine-tuning with ESC-50. We also freeze 15 layers, half of the model, since we can not conclude the optimal frozen layers to retain the performance of Wav2Vec's specialized task. We closely observe the training and validation loss during the process and record the accuracy. Since AudioSet is a larger dataset, we focused on three parameters in the model to improve the performance of WAv2Vec on AudioSet. First, we focused on increasing the number of layers in the classifier architecture to have more learning parameters for the larger training data. We add a dense layer with the parameters of 2048 to 1024 between the original dense layers used in Section 3.1.1.4 and change the output of the first layer to 2048 and the input of the last layer to 1024. Increasing the complexity of the model also enables the classifier to learn more complex patterns but increases the risk of overfitting the model.

Based on the observation we find from the training and validation loss and our anticipation for overfitting, we explore a couple of techniques that will prevent overfitting. The first technique is L2 regularization. L2 Regularization is a technique that takes the sum of all the weights squared to the loss function. This reduces the magnitude of weights, preventing it from growing too large. Within our code, we add this to the optimizer function via a weight decay parameter. We test 0.0001 as our weight decay for the L2 regularization. In addition, dropout layers are another method we investigated, where dropout layers are placed typically after dense or convolutional layers. Dropout layers exclude a percentage of neurons in the layer before the dropout layer during a training run. Since dropout layers are already used in the classifier model, we want to see the effects of Wav2Vec's performance working without it. After we observe the results of each fine-tuning step, we combine the best-performing parameters and save the model after training with AudioSet. Since the training process with AudioSet is very time-consuming, we decide to focus our attention on L2 regularization. L2 regularization presents itself as the best candidate for more improvement as it shows the most promising results, so we decide to explore L2 regularization weight decays further. We observe the effects of 0.0001, 0.001, and 0.01 weight decays on the performance of the model. We will also save the best model for this and record the accuracy as well. With the saved Wav2Vec model trained on AudioSet, we use transfer learning to test the saved audio embeddings with ESC-50 data and observe its accuracy performance.

3.2.4 Fine-tuning with Jialu et al.'s [16] Methodology

Wav2Vec2. In order to utilize Jialu et al.'s [16] methodology, we adapt code from the paper's GitHub written by Yuan Gong, one of the authors. This code consists of 2 files: one file, which handles feature extraction of the unlabeled audio data, and one file, which classifies the extracted features and trains the embedding based on the cross-entropy of the classifications. In the feature extraction file, we load the Wav2Vec2 model "facebook/wav2vec2-base-960h" along with its processor. Yuan's pipeline then splits the data into five folds and extracts the audio features, parsing both train and evaluation data into JSON files. Following this, in the classification file, we once again load the model and its corresponding processor. Yuan's pipeline then clusters the extracted features for training and evaluation data in each of the folds. Afterward, the clustered data is fit to each of the model's embedding layers, the output of which is passed to an MLPClassifier. This pipeline outputs a prediction for the evaluation data, which is used to calculate accuracy and F1 metrics for each layer of the embedding over all the folds. The metrics for all layers of the embedding are output to a csv file [16]. We adapt this code to find the layer with the best performance, and then we save the trained model embedding as a .pt file.

Wav2Vec. Jialu et al [16] methodology also has potential applications to Wav2Vec by taking advantage of Wav2Vec's 20 layers of feature aggregators after the 10 layers of feature extractors. The data pre-processing steps, classification steps, and sources we use are exactly the same as the procedures stated for Wav2Vec2's Jialu's method. However, where it differs is in the feature extractor file. Wav2Vec is loaded through Fairseq and is not supported by Huggingface as Wav2Vec is considered obsolete. This limits us from being able to use Huggingface's function to attain the outputs of every layer in the model, also known as hidden states. Wav2Vec2's architecture has a section where the layers will output the same size tensor for it to be clustered. Consequently, the size of each feature extractor layer output is not consistent, so we use the feature aggregator outputs since they are the same size tensors. This prompts us to create a function that will record the outputs of just the feature aggregate outputs and cluster it for the classifier file. This allows Jialu et al methods to be used on Wav2Vec, so we can observe the results and save the model.

3.2.5 Wav2Vec ESC-50 Fine-tuning Results and Discussions

In this section, we outline the results gathered from Section 3.2.2. The initial test for an increased number of epochs to 60 shows positive results of upwards of 7% increase in accuracy. This prompts all tests to be done with 60 epochs.

The investigation into the optimal learning rate for Wav2Vec yields interesting results, which are

presented in the graph shown in Figure 1. The results indicate that a constant learning rate of 1e-4 performs the best, while decreased learning rates result in a decrease in performance. This trend can be attributed to the fact that the lower learning rate is too small and does not allow the model to reach the lowest point of the loss function fast enough, resulting in slower convergence and sub-optimal performance. This trend is also consistent for 27 and 25 frozen layers, showing that the number of frozen layers and the learning rate are independent.

Figure 1: Wav2Vec Performance on Different Learning Rates Using ESC-50

Further investigation into different learning rate techniques, like the results of exponential decay, are shown in figure 2. The results indicate that an exponential decay factor of 0.9 is the best performing, while higher or lower decay rates yield bad performance. Lower exponential decay factors are too aggressive in their decrease in learning rates, thus not allowing the model to converge enough. The higher exponential decay rate makes too little of a change in the learning rate to impact performance significantly and is similar to results found in Section 3.1.2 with ESC-50 on Wav2Vec.

Finally, 3 shows promising results with performance increasing to over 50% accuracy, using the linear step decay technique. The results indicate that the optimal learning rate is achieved using an initial learning rate of 1e-4, with a step size of 10 epochs and a multiplicative factor of 0.9 to the learning rate.

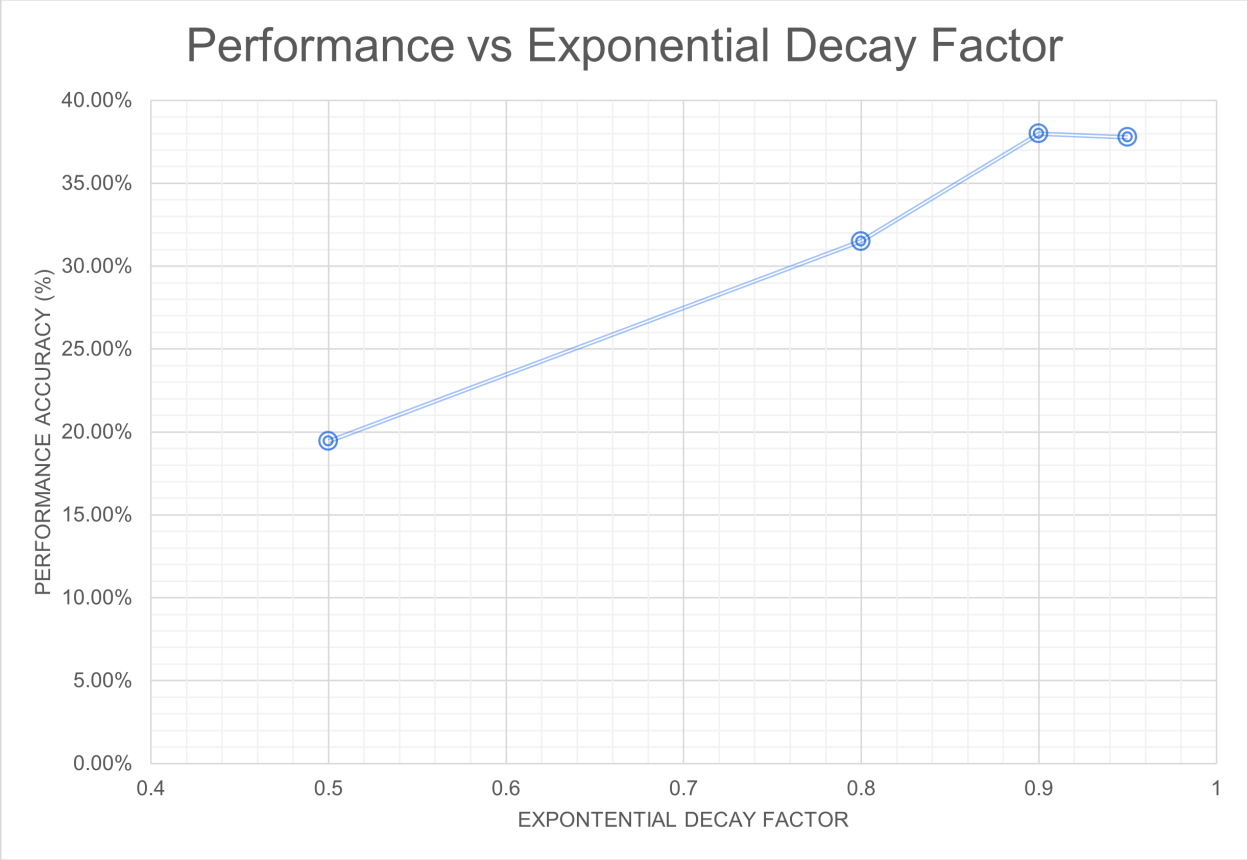

Figure 2: Wav2Vec Performance on Different Exponential Decay Rates Using ESC-50

This yields a 51.86% accuracy to the model. The graph also shows a different number of epochs it goes through before it applies the decay, that being 10 and 20 epochs. For 10 and 20 epochs, it shows the lowest being 26.83% and 37.68% with the lowest multiplicative of 0.1 and the highest at 51.86% and 50.89% with the highest multiplicative of 0.9, respectively. The general trend shows that the increased multiplicative factor to the learning rate will result in better performance on the model for non-speech tasks. This is due to lower multiplicative factors being too aggressive in decreasing the learning rate and negatively impacting the model's convergence. The change in step sizes of epochs shows that 20 epochs are more resilient to change in the learning rate but is not the best performer when the optimal multiplicative factor is found. This is because the learning rate changes less often when the step size increases. However, when observing 10 epochs, the results show higher sensitivity to the step size since the learning rate changes more often and shows the worst performance with an unfit learning rate multiplicative but the best results for optimal learning rate multiplicative.

Based on the graph presented in figure 4, it can be observed that the number of frozen layers in the Wav2Vec audio embedding has an impact on the model's performance. As the number of frozen

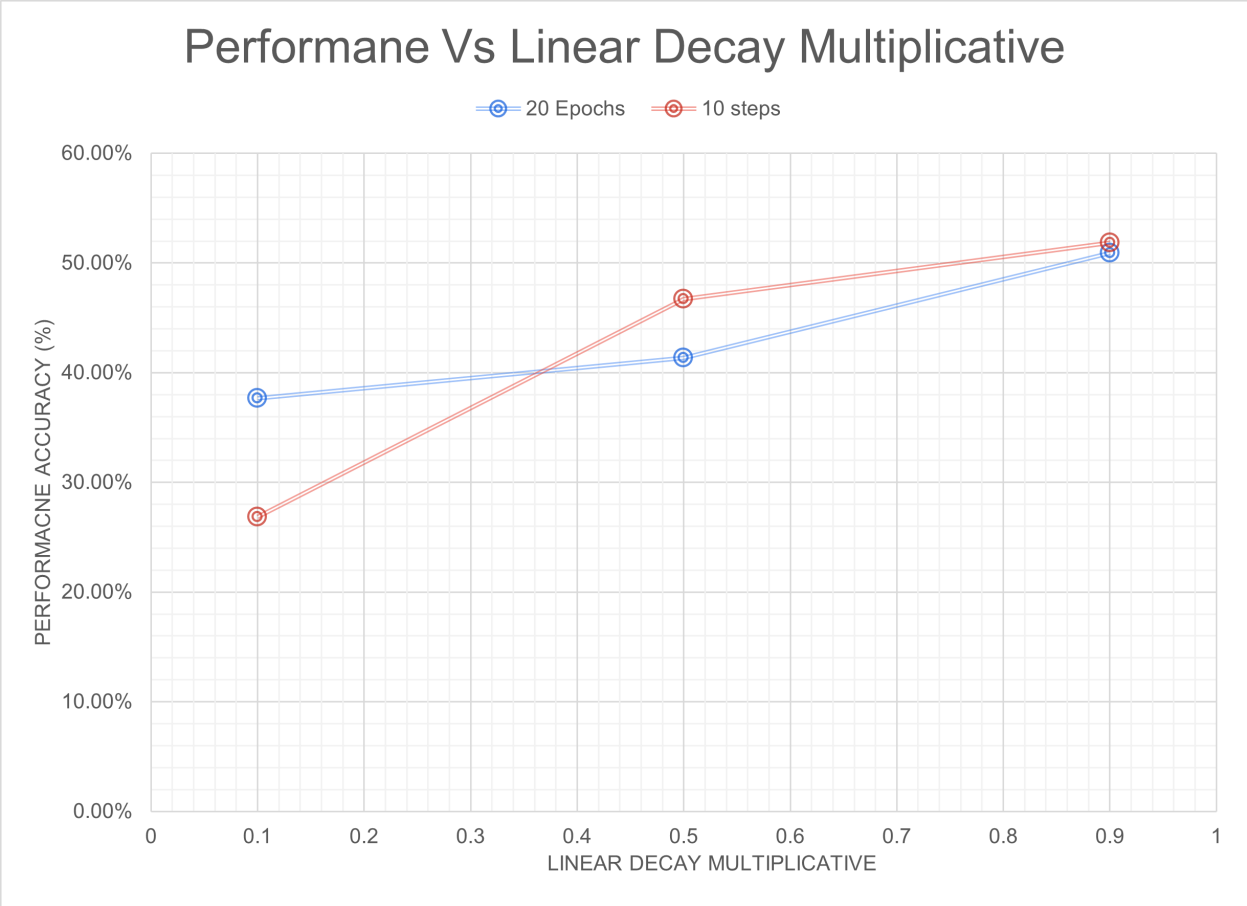

Figure 3: Wav2Vec Performance on Different Linear Decay Rates Using ESC-50

layers decreases, the Wav2Vec model will perform better as there are more trainable parameters to learn environmental sound features. The best accuracy of 51.86% is obtained when no layers are frozen, and the worst performance is with 25 layers frozen, resulting in an accuracy of 35.88%. It is also important to mention that some data points don't follow the general trend, such as 15 and 30 layers frozen, due to the unpredictable nature of training machine learning models. One idea of this phenomenon is the randomly initialized weights in the classifier model. It is possible that the weights are coincidentally close to the optimal weights for the ESC-50 dataset, resulting in better accuracy even with a higher number of frozen layers. Conversely, the weights can also be coincidentally close to the worst weights for this dataset, causing a lower accuracy with a lower number of frozen layers. These nuances make it challenging to predict the exact performance of the model with a specific number of frozen layers.

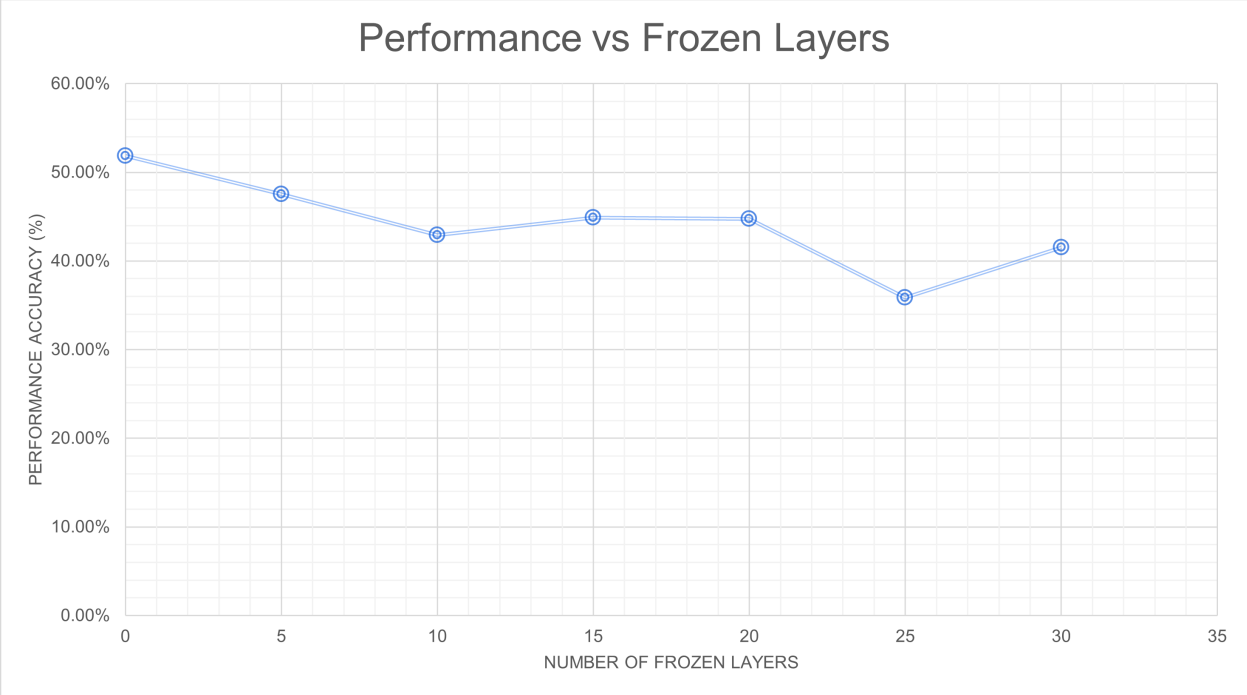

Figure 4: Wav2Vec Performance on Different Number of Frozen Layers Using ESC-50

3.2.6 Wav2Vec AudioSet Fine-Tuning Results and Discussions

The results of the procedures mentioned in Section 3.2.3 can be seen in both figures 5 and 6.

Figure 5 shows the different fine-tuning techniques' impact on performance with Wav2Vec on AudioSet. Regular uses all best-performing fine-tuning techniques for hyperparameters like the dynamic learning rate and number of epochs from Section 3.2.5 but for AudioSet. These hyperparameters show a 0.4% increase in performance from table 4, which can be considered very positive due to the difficulties of classifying 527 unique classes. However, we observe the validation loss increase throughout the training process after 7 epochs, which strongly indicates overfitting. This phenomenon explains the importance of the dropout layers in the classifier model, as without the dropout layer, there is a significant loss in accuracy, which results in 0.08%. Additionally, the problem with overfitting is evident when applying L2 regularization, a method of preventing overfitting, and it improves accuracy by 0.2% from the regular. During this training process, the validation loss continues to decrease till 30 epochs in before increasing again. We also discovered that having more parameters in the classifier model has also increased the accuracy by 0.1% from the regular. We believe that an increase in parameters makes the classifier model able to learn more complex tasks, especially for AudioSet's immense number of data samples and classes.

From the results we gather from figure 5, we conclude that having the hyperparameters from the

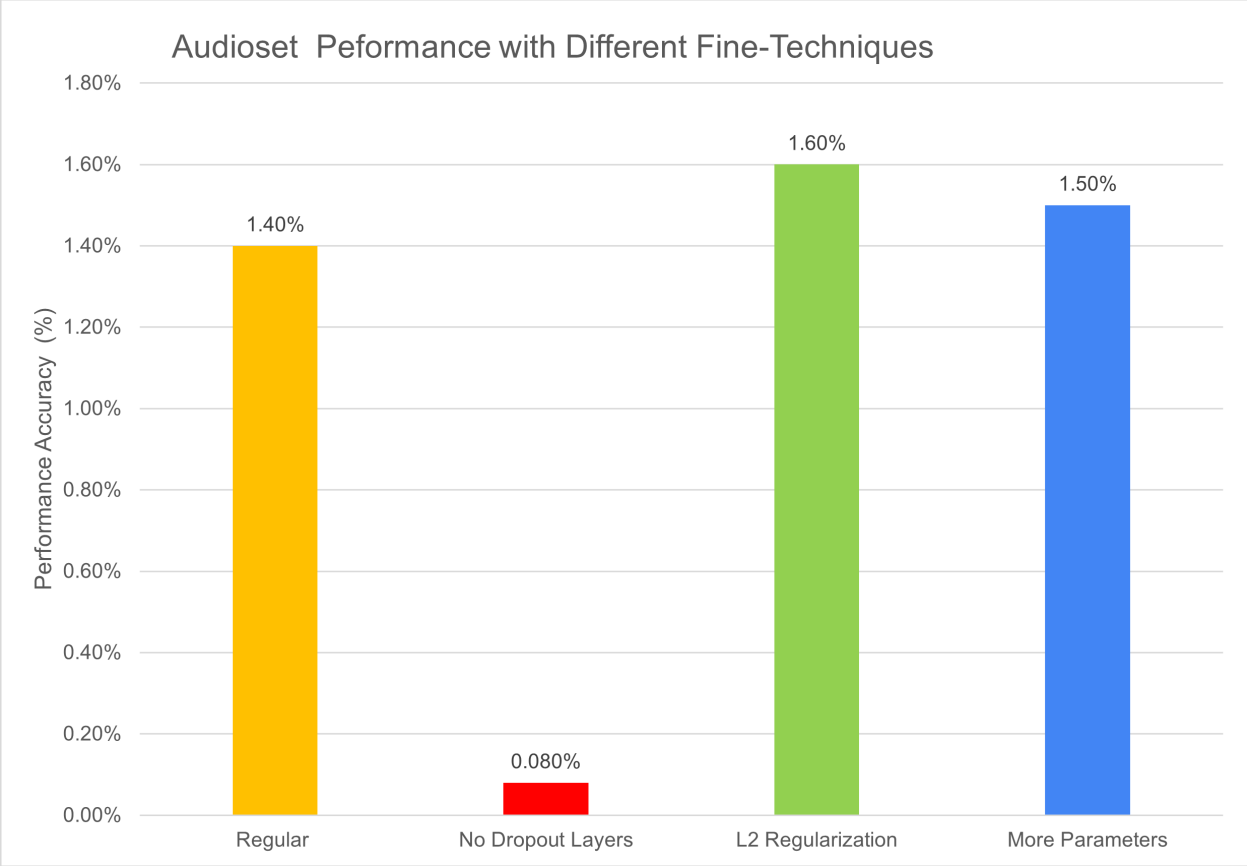

Figure 5: Wav2Vec Performance on Different Fine-Tuning Techniques Using AudioSet

regular, keeping dropout layers, introducing L2 regularization, and having more parameters in the classifier model is the best-performing model for Wav2Vec on AudioSet. This combination of fine-tuning techniques has resulted in 1.80%, which is a 0.2% increase from all the results from figure 5. The results produced from different L2 regularization weight decays are shown in figure 6. The graph shows that 0.001 weight decay is the most optimal as it causes the performance to increase to 2.20%, double the inference results from Section 3.1.2. Anything less than 0.001 indicates that the weights of the trainable parameters are becoming too large, thus overfitting. On the other hand, when the weight decay is higher than 0.001, it suggests that the weight decay is penalizing the weight too harshly, making the model underfit.

We apply transfer learning techniques with the best Wav2Vec on AudioSet model embedding (2.20% accuracy) to Wav2Vec on ESC-50. The result is underwhelming, as we discover that the model achieves 35.46% with ESC-50. This metric is similar to the inference result we gathered from Wav2Vec on ESC-50. We believe that the AudioSet data is too large and complex for half of Wav2Vec's trainable parameters to learn, not to mention that AudioSet also includes speaker diarisation tasks in its dataset. Overall, using transfer learning and fine-tuning Wav2Vec with AudioSet is not feasible.

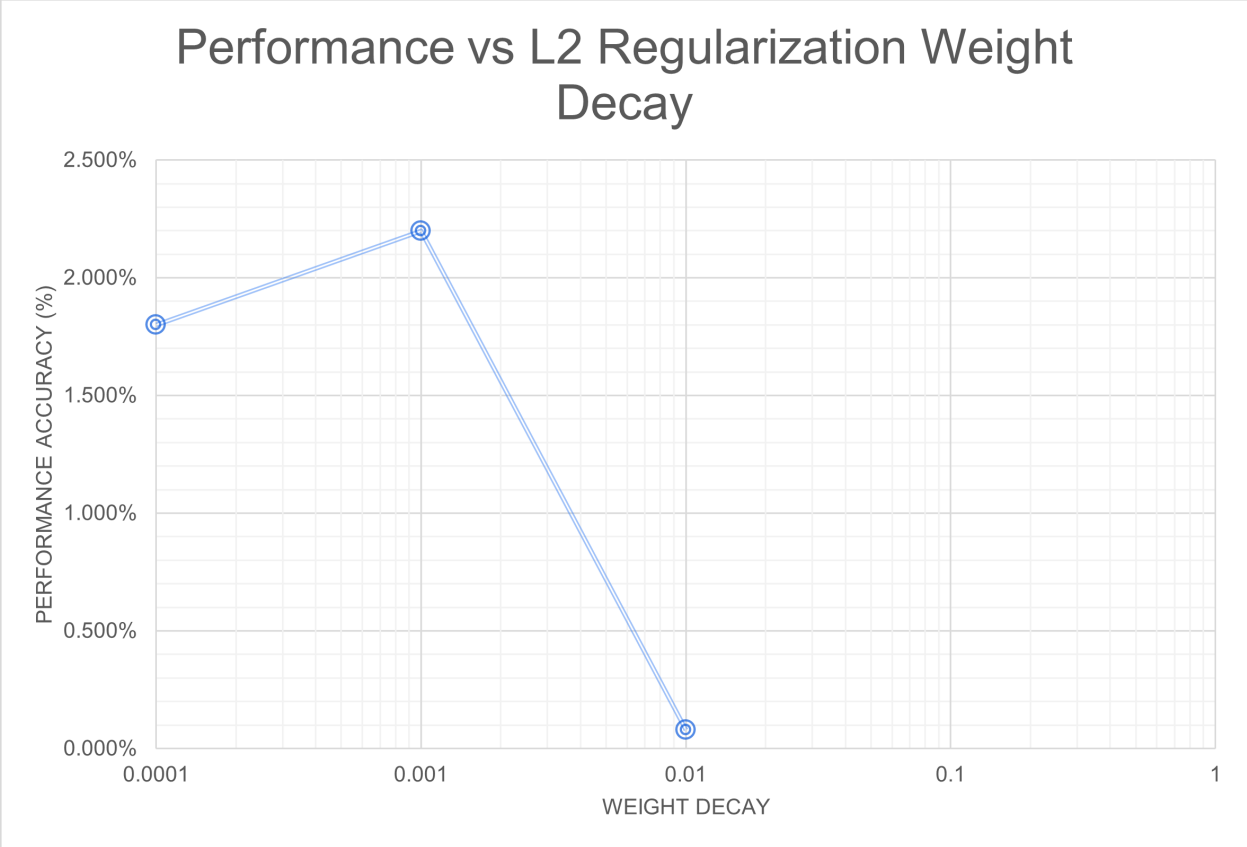

Figure 6: Wav2Vec Performance on Different L2 Regularization Weight Decays Using AudioSet

3.2.7 Fine-tuning Results Utilizing Jialu et al.'s[16] Methodology

Wav2Vec2. The results of the procedures mentioned in Section 3.2.4 can be seen in figure 7. This figure represents the F1-scores achieved by each layer of the Wav2Vec2-base-960h model using this methodology. As can be seen in the figure, the best-performing layer is layer one with an F1-score of 0.66. The worst layer is layer 11, reaching 0.33. Lastly, the output layer, 12, achieves 0.62. Based on these results, we can gather that fine-tuning the full Wav2Vec2 model has the most potential as a combined embedding for speech and non-speech tasks. While layer one does outperform layer 12 by 4%, using only the second layer would lead to too great a degradation in Wav2Vec2's ability to perform speech tasks. Therefore, maintaining all layers of the model is essential.

While we train on the full embedding to achieve an output F1-score of 0.62, it is unclear how this process affects the effectiveness of the model when performing human speech tasks, such as when comparing to the ASR scores in table 5. While we are not successful in performing ASR inference with this fine-tuned model on the LibriSpeech dataset, future studies should observe this result as a means of qualifying the

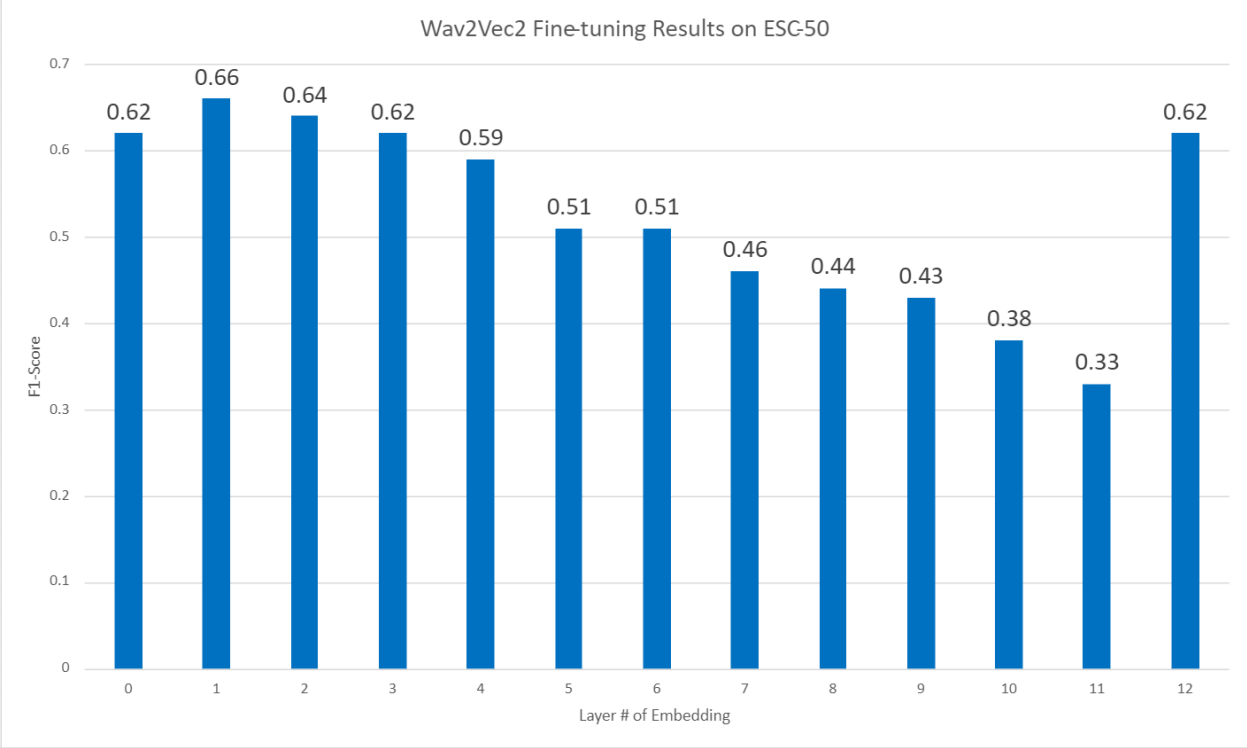

Figure 7: Wav2Vec2 F1-score results after fine-tuning on ESC-50

model's speech task capability. Moreover, we do not investigate the effect of freezing individual layers when fine-tuning using Jialu et al.'s [16] algorithm. A future study might investigate this as a way to prevent degradation of the fine-tuned model's speech task capability whilst increasing the capability of non-speech task via the fine-tuned layers, as is explored in Section 3.2.2.

Wav2Vec. Wav2Vec's results using Jialu et al.'s [16] pipeline consistently produce an average accuracy of 3% for each fold. After discussing this with our Ph.D. student, we determined the results are invalid due to the significant difference in the Wav2Vec and Wav2Vec2 model architecture. This will be described in further detail in Section 4.3.

3.3 Inference for Alternative Speech Tasks

Following our conclusions drawn in Section 3.1.2, we realize that the common human speech task, ASR, is not suited to the AST model. The AST model is designed to intake audio clips of fixed length [4], whilst ASR is performed on clips of variable length when using the LibriSpeech dataset. While this does not invalidate our fine-tuning results in Section 3.2.5, it does warrant further study into the capabilities of AST for alternative human speech tasks.

In order to evaluate the effectiveness of AST for alternative human speech tasks, we aim to perform inference for a speaker recognition task using the Vox Celeb1 dataset. We prepare the data for inference by taking 12 audio clips from each speaker in the dataset. Of these 12 audio clips, 6 are used to form 3 pairs of same-speaker audio clips, while the other 6 are paired off for 3 pairs of different-speaker audio clips. All 7506 pairs are placed into independent folders labeled from 1 to 7506. As pairs are formed, a csv is written containing the ID of the speaker of each clip in the pair, the folder index of the pair, and a "same_id" label. If the speaker for both clips in the pair is the same, this label is 1. Else, the label is 0. We encounter an error where the names of the audio clips in the new pair folders are the same since they are numbered independently for each speaker. In order to avoid OS errors with the file system, we fix this by tacking "_01" and "_02" onto the end of the name of each audio file in a pair.

After audio clip pairs are formed, the code for data preparation is handed off to the Ph.D. student, who will use it to train a classifier and perform inference on the Vox Celeb1 dataset with an Equal Error Rate (EER) as the success metric. For comparison to other models, inference on the Vox Celeb1 dataset already has state-of-the-art results on Wav2vec2 and is explored by the Ph.D. student on Wav2vec. Current work on this project handled by the Ph.D student also explores the effectiveness of the AST, Wav2vec, and Wav2vec2 models on stutter detection datasets. Comparing the performance of these models on both stutter detection tasks and speaker recognition tasks will allow future research on this project to determine the capability of AST for alternative human speech tasks and thus reassess its viability as a combined embedding for human speech and non-speech acoustic tasks.

4 Limitations and Future Works

Our team encountered several limiting factors throughout this study. This section provides an in-depth analysis of these limitations, beginning with a brief talk about available computing resources. Following this, we discuss the compatibility of the ASR task with the AST model and touch on the limitations of Jialu et al.'s [16] method with the Wav2vec model's architecture. Finally, we conclude with limitations encountered due to limited time and period of study, which include the inability to explore layer freezing for the Wav2Vec2 model when fine-tuned using Jialu et al.'s algorithm, and similar hindrance in performing ASR inference on our fine-tuned models.

4.1 Limited Compute Resources

The compute resources available to our team slowed us down most significantly while performing inference on the AST model. Processing power and memory overhead on personal devices and Google Colab were too limited to perform any inference tasks on this model, leading us to hand prepared data off to a Ph.D. student with access to compute resources on the WPI Turing Cluster. While this did not prevent us from completing inference for these models, it did hinder progress for a notable period of time, as we originally attempted to perform inference using local and cloud GPUs before performing this hand-off.

4.2 AST Model's Compatibility with ASR Task

One of the largest limitations our team encountered is the suitability of the AST model when performing ASR inference on the Librispeech 100-hour dataset. The AST model is designed to intake audio clips of fixed length, thus making it unsuitable for processing the Librispeech dataset [4]. We were unaware of this while utilizing our inference results to inform our team of which model to fine-tune and observe further. It's important to note that this limitation of the AST model does not invalidate any of our fine-tuning results for the Wav2Vec and Wav2Vec2 models. It does prompt further observation of the potential of the AST model as a dual-purpose embedding, however, given our initial write-off of the AST model when viewing the inference results in section 3.1.2. This realization prompts our team to begin performing inference for alternative audio tasks, which will be essential to the future exploration of AST's suitability for human speech acoustic tasks, which will be explored in the future of the project.

4.3 Jialu et al.'s[16] Algorithm Limitations on Wav2vec Architecture

As discussed in section 3.2.7, Jialu et al. [16] have successfully implemented a method to improve Wav2Vec2's performance with the ESC-50 dataset. However, as per table 3, Wav2Vec2 has the largest parameter size among the three options discussed. It requires almost 1 GB of storage, which is not feasible for most affordable embedded devices, making it impractical for real-world applications. While Wav2Vec seems to be the best option in terms of memory, performance, and architecture, we discovered that Wav2Vec's architecture [3] is different from Wav2Vec2 [2]. The features that we are clustering from Wav2Vec2 are from its 12 encoder layers, while for Wav2Vec, we take the features from the 11 layers of the feature aggregator. These encoder layers and feature aggregators have different purposes and output different meaningful representations that can either be useful or useless for Jialu et al.'s method. Therefore, Jialu et al.'s practicality

in future works with audio embedding other than Wav2Vec2 is limited for further exploration.

4.4 Layer Freezing for Wav2Vec2 Fine-tuning

In section 3.2.2, we discuss layer freezing as a means of preventing degradation of human speech acoustic task performance on the Wav2vec model when fine-tuning on the ESC-50 dataset for non-speech acoustic tasks. However, due to limitations in time, as we neared the end of our study, we were unable to explore the effects of layer freezing when fine-tuning the Wav2Vec2 model using Jialu et al.'s [16] methodology. Just as with the Wav2vec model, a study in this direction is important for maintaining Wav2Vec2's human speech task performance when fine-tuning for non-speech acoustic tasks using Jialu et al.'s algorithm. This is essential for creating an effective dual-embedding for both speech and non-speech tasks. In the future, we hope to add an algorithm to freeze some of the layers in the Wav2Vec2 model with Jialu et al.'s method.

4.5 ASR Inference on Fine-tuned Models

With the combination of all the challenges we encountered with our studies, the processing times to train and validate different fine-tuning techniques, and the limited bandwidth we had during our studies, there is a crucial step that we did not get to explore. This is testing the performance of all the models on speech-related tasks, explicitly evaluating the models on the LibriSpeech dataset. More specifically, the difficulty lies in the library, fairseq, which we used to load Wav2Vec with, since Wav2Vec is not supported by HuggingFace. Fairseq has minimal functions, and one of the critical functions we needed is the decoder function, which exists for Wav2Vec2 on HuggingFace. This decoder function converts the outputs of Wav2Vec to predicted words so that the word error rate can be calculated. Without the decoder, the function must be written ourselves, which we do not have the bandwidth for. As a result, the observations made in sections 3.2.5, 3.2.3, and 3.2.7, can not be established as success or not, without the evaluation of the speech-related tasks. In the future, we hope to have the performances of our fine-tuned models with speech-related tasks to determine the degradation of our models as well as their practicalities.

5 Conclusion

Through this project, we show potential ideas and lay the foundations for creating a single audio embedding capable of doing both speech and non-speech classifications for an embedded system. Specifically, our study establishes the potential of the Wav2vec and Wav2Vec2 models as combined speech and non-speech

embeddings through our exploration of inference on multiple environmental sound classification datasets. Our study then yields fine-tuning results for these potential audio embedding candidates, the best of which were Jialu et al.'s method with Wav2Vec2 (62%) and carefully selected fine-tuning techniques with Wav2Vec (51.86%). Although these models are far from being perfect and need further evaluation, our results are a step in the right direction toward our goal of this project.

Future applications of this project include new opportunities for single audio embeddings for speech and non-speech tasks, such as development for deployment on embedded systems like the smart home device. This can be expanded into the healthcare field, such as hearing aids capable of classifying different sounds regardless of whether they are speech or non-speech sounds. This research can also be used for combined speech and audio tagging on autonomous home security systems. The potential applications of this project emphasize the importance of further research and development for multifaceted single audio embeddings.

References

[1] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022.

[2] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[3] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," 2019.

[4] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," 2021.

[5] Y. Gong, S. Khurana, L. Karlinsky, and J. Glass, "Whisper-AT: Noise-robust automatic speech recognizers are also strong general audio event taggers," in *INTERSPEECH 2023*, ISCA, aug 2023.

[6] P. Mohapatra, B. Islam, M. T. Islam, R. Jiao, and Q. Zhu, "Efficient stuttering event detection using siamese networks," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[7] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM International Conference on Multimedia (ACM-MM'14)*, (Orlando, FL, USA), pp. 1041–1044, Nov. 2014.

[8] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[9] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018, ACM Press.

[10] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, (New Orleans, LA), 2017.

[11] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017.

[12] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015.

[13] "What consumers really think about ai: A global study." https://www.pega.com/ai-survey/.

[14] "Market research report."

[15] T. Mariotti, "Smart home statistics (2024)," Aug 2023.

[16] J. Li, M. Hasegawa-Johnson, and N. L. McElwain, "Towards robust family-infant audio analysis based on unsupervised pretraining of wav2vec 2.0 on large-scale unlabeled family audio," in *INTERSPEECH 2023*, interspeech$_2$023, $ISCA$, Aug.2023.

[17] J.-C. Wang, H.-P. Lee, J.-F. Wang, and C.-B. Lin, "Robust environmental sound recognition for home automation," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 1, pp. 25–31, 2008.

[18] "Smart home security market by component (hardware, software, service), by device type (smart alarms, smart locks, smart sensors and detectors, smart camera and monitoring system, other) and region, global trends and forecast from 2024 to 2030," Feb 2024.

[19] "The evolution of text embeddings," Oct 2023.

[20] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, p. 279–283, Mar. 2017.

[21] J. Barker, S. Watanabe, E. Vincent, and J. Trmal, "The fifth 'chime' speech separation and recognition challenge: Dataset, task and baselines," 2018.