Environmental Sound Classification using One/Few Shot Learning with Siamese Networks

by

Seyed Ashkan Tavassoli Kakhki

B.Sc., Sharif University of Technology, 2017

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Mechatronic Systems Engineering
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2022

Declaration of Committee

Name:	**Seyed Ashkan Tavassoli Kakhki**
Degree:	**Master of Applied Science**
Title:	**Environmental Sound Classification using One/Few Shot Learning with Siamese Networks**
Committee:	**Chair:** **Ahad Armin** Lecturer, Mechatronic Systems Engineering
	Siamak Arzanpour Co-supervisor Associate Professor, Mechatronic Systems Engineering
	Elina Birmingham Co-supervisor Associate Professor, Education
	Mohammad Narimani Committee Member Lecturer, Mechatronic Systems Engineering
	Amir Shabani Examiner Lecturer, Sustainable Energy Engineering

Abstract

The inability to tolerate everyday sounds, also known as Decreased Sound Tolerance (DST), has proven to be one the most prevalent issues in Autism Spectrum Disorder (ASD). While advanced Neural Networks have shown promising results in classifying environmental sounds, those conventional classification models rely on sound classes that were used in the training process. In DST, the list of aversive sound classes may be unique and different for each person, and training a conventional classification model that can classify *all* possible aversive sound classes is not feasible. Hence, a classification approach that works beyond this limitation is required. In this thesis, the idea of One/Few Shot Learning for environmental sound classification is explored. This model can classify a given sound by having one or very few samples of that class. As a part of this research, different aspects of the model are optimized and a state-of-the-art model is developed.

Keywords: Autism Spectrum Disorder (ASD); Deep Learning; One/Few Shot Learning; Environmental Sound Identification; Sound Classification; Transfer Learning

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Dr. Arzanpour and Dr. Birmingham, for their invaluable patience and feedback. Without them I would not have made it through my master's degree.

I'm extremely grateful to my parents, Mehri and Mehdi, for their unconditional love and support and a special thanks to my one and only brother, Arash, who helped me throughout the entire thesis process and every day of my master's program.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ASD	Autism Spectrum Disorder
DST	Decreased Sound Tolerance
ANN	Artificial Neural Network
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
ESC	Environmental Sound Classification
LSTM	Long-Short Term Memory
STFT	Short-Time Fourier Transform

Chapter 1. Introduction

1.1. Background and Motivation

Autism Spectrum Disorder (ASD) is a neurodevelopmental disorder and characterized by social communication impairments, and the existence of restricted interest and repetitive behaviors [1]. As estimated by the Autism and Developmental Disabilities Monitoring Network (ADDM), the prevalence of ASD in the US has more than doubled between 2000-2002 (one in 150 children) and 2010-2012 (one in 68 children) [2]. Sensory processing issues have been frequently reported in ASD despite the general heterogeneity of the disorder. Notable sensory processing issues have been reported by 90% of autistic individuals [3].

One of the most persistent and disabling sensory processing issues in autism is the inability to tolerate everyday sounds, called Decreased Sound Tolerance (DST) [4][5]. Based on a recent meta-analysis estimation by Williams et al. [6] 50-70% of autistic individuals have experienced DST at some point in their lives and the current prevalence of DST in the autistic population is estimated to be between 38% to 45%. Many caregivers have reported that DST prevented the children from participating in many situations ranging from family to community activities [7]. For autistic individuals, certain sounds trigger extreme aversive reactions that affect their everyday life. Although the nature of these sounds varies between different individuals, most caregivers have described these sounds as loud, sudden, and high-pitched [8].

Disorders of DST can be defined in 3 specific conditions: hyperacusis, misophonia, and phonophobia. **Hyperacusis** is a hearing disorder characterized by decreased tolerance of sounds at a level that would not trouble others. Everyday sounds can become unpleasant, painful or overwhelming for someone with hyperacusis, where these sounds are often perceived as louder compared to how a non-autistic individual would perceive them [9]. **Misophonia**, a newly described neuropsychiatric condition, is characterized by inappropriate and excessive emotional responses to specific trigger sounds such as chewing or tapping, even at low amplitudes. The triggers and responses vary greatly between different individuals, and they may change over time, however, primary responses include some form of anger, irritation, and disgust [10]. **Phonophobia**, which translates to fear of sound, is a common term in neurology to

describe sound intolerance typically accompanied by migraine headaches [11]. In this framework, phonophobia is defined as the fear of specific sound classes that would result in anticipatory responses and avoidance of prospective sound sources [12][13].

Common methods used by caregivers to help with the DST disorders in autistic people, such as warning or avoiding noisy settings [8], removes the individual from the environment and causes social exclusion. Wearable devices like earmuffs and noise-canceling headphones can block all the ambient sounds including the aversive ones, but these approaches isolate the individual from the environment by blocking sounds such as human speech which can also result in social exclusion.

To create a method that helps with DST but does not result in social exclusion, an intelligent system acting based on each individual's preferences is required since sound sensitivity in autistic individuals is a subjective issue and the list of aversive sounds can vary from one person to another [8]. To create an intelligent and personalized device that finds and blocks these aversive sounds, the system should be able to accurately classify different ambient sounds and block only those which are aversive for that specific individual. The intelligent system can be separated into two major blocks, a sound classification and detection block and an intervention block which would be activated when an aversive sound is detected by the classification.

The intervention block is out of scope for this thesis but in general the intervention can happen in different manners such as filtering the aversive sound, playing white noise to cover the aversive sound, or blocking all the sounds. For the sound classification and detection block, machine learning models have shown high accuracy in classifying environmental sounds. Machine learning models can vary from classic machine learning algorithms to a much more complex approach called the Deep Neural Networks (DNN)[1], but for environmental sound classification DNN models have a much higher accuracy in comparison with the classic methods [14], [15]. However, in the common DNN model training approaches the model learns to identify separate classes based on the knowledge learned from the training data (DNN models learn to solve a problem by analyzing a provided dataset called the training data). This results in a model that can only detect and classify sound classes that were used in the training process (**seen**

[1] Explained in Chapter 2

classes) and is unable to classify a sound class that did not have any samples in the training data (**unseen classes)**. To address the issue of DST in autistic individuals, the system must be able to detect and classify all seen and unseen sound classes, which would be impossible using such approaches. Common aversive sounds can be used to train a model and then be classified by such models with high accuracy, but an alternative approach that can work beyond the classification limits of a DNN model would be necessary to work alongside the high-performance classification model and detect the sound classes that are not in the classification model's vocabulary.

As an alternative to traditional classification approaches, One/Few Shot Learning can be used to create a classification model that is not limited just to the training data. The idea of One/Few-Shot Learning is to create a model that can compare two inputs and find the probability of them being in the same class [16]. Since the One/Few-Shot Learning is trying to be a general answer and not dependent on the training classes, a successful implementation should be able to compare an input with any class by just having one/few samples of that class and hence, classify an input into any classes that it has sample/samples of. This idea has shown to be greatly powerful in the vision domain, such as image recognition [17] and face recognition [18], but the use of One/Few-Shot Learning in audio detection or classification has not been intensely explored.

1.2. Previous Work

Over the past decades, audio classification has been a subject undergoing intense study in machine learning due to its impact on how machines can interpret their environment and help in various fields such as hearing aids [19], surveillance [20], and even disease identification (e.g., lung sound classification [21]). In general, a classification approach in any area such as sound or image would only be able to classify an input to the classes that were used in the training process. The idea of One/Few Shot Learning was introduced to address the limitation caused by being only able to classify and compare the classes used in the training process [16] [22]. The approach is defined as a Bayesian Model[2] which can generalize into new unseen

[2] A model where inference is based on using Bayes theorem to obtain a posterior distribution using prior distribution for the relevant parameters

classes with one or very few samples, called **One Shot Learning** for one sample and **Few Shot Learning** for few samples.

The One/Few Shot Learning idea is a general approach that can be applied to many different fields, but it has been mostly investigated in computer vision. Many successful implementations have been developed to address image classification and object detection problems [23][24][25], with the help of large image data sets such as ImageNet [26] and Omniglot [27]. One/Few Shot Learning has also shown promising results for face recognition problems [28]. In audio-related applications of One/Few Shot Learning, improvements have been made on speaker identification [29][30] and there have been studies on audio classification using a small labeled dataset (Few Shot Learning) where the authors reported an accuracy of ~74% on UrbanSound8k [31] dataset [32] (a audio dataset with ~8k labeled sounds of 10 urban sound classes). In a newer study, using attentional graph neural networks, a 5-way 5-shot classification (a classification with 5 samples per class and 5 classes) was done with the highest accuracy of %78.3 [33].

An effective approach for One/Few Shot Learning is called Siamese Networks [17], a subset of Deep Neural Networks[3] with two identical sub-networks having shared weights (called the **feature extractor)**, and a comparison function that compares the outcome of these two identical networks[4]. The idea here is to use two distinct inputs and find the probability of these two distinct inputs belonging to the same class (or having the same origin). Siamese Networks were originally introduced in 1994 for signature verification by calculating a similarity score between two inputs [34].

The most important step for creating a Siamese Network for One/Few Shot Learning is defining the feature extractor. An optimized feature extractor for the Siamese Network can be the *feature extractor* component of a classification model trained to solve a similar problem (like audio classification). We know that In general, each conventional classification model is comprised of: (i) a *feature extractor* and (ii) a *classifier.* Therefore by dropping the last layers which act as the classifier we can use the rest as the network's feature extractor. Even though the difference between an

[3] Explained in Chapter 2

[4] Further explained in 2.4

image and a spectrogram[5] of audio (most common representation of an audio for sound classification) is significant, image classification model architectures have achieved state-of-the-art results in sound classification [35] and can be considered as a possible feature extractor.

To train the feature extractor, one can either start with random values for the initial weights (starting point of training) or use a method that expedites and enhances the training process called transfer learning, a method where a model trained on a large dataset for a particular task is extended to another task (usually a similar one) and learns to solve the second task based on prior knowledge. Transfer learning has been used in many cases to improve a classification model's accuracy. Examples of transfer learning from image datasets such as ImageNet have proven to be effective for tasks like image segmentation [36] and medical image analysis [37]. Transfer learning has also been widely applied in audio-related tasks, where models trained on AudioSet [38] and Million Song Dataset [39] have shown a performance improvement for multiple applications [40][41]. The idea of applying transfer learning to image classification models trained on image datasets and using that for audio classification problems has been around for a while [42] and transferring knowledge from a model pre-trained on ImageNet has been shown to improve the performance of multiple audio classification and detection problems [43][44][45].

The feature extractor, with or without transfer learning, would be able to extract sound features from one frame (window in time) but extracting features that can fully represent a sound may require looking at more than one frame. Splitting a sound into multiple frames in time can show the feature extractor how the sound has changed over time as well as the features of the sound in each frame – which can lead to a more accurate representation of the sound. Hann Window, a renowned signal processing method invented by Julius Von Hann around 1900, is used to reduce the importance of borders when windowing a signal such as a sound. Using this method, audio could become a series of frames showing how the sound has changed over time. Although looking at all these frames at the same time is a possibility, finding a compact representation of them may be needed to improve a One/Few Shot Learning model.

[5] Further explained in 3.1.1

Auto-Encoders[6] were introduced to solve a similar task - training an *encoder* that finds an internal representation of an input and at the same time training a *decoder* that reconstructs the input from this internal representation [46]. In case of One/Few Shot Learning, the idea is to use the *encoder* components of such systems to build a compact representation of multiple frames of an input sound (as a second layer of feature extraction)[7]. A more recent version of Auto-Encoders using Recurrent Neural Networks (RNN)[8] could find a representation of sequential data effectively [47]. LSTM[9] (Long Short-Term Memory, a type of RNN immune to exploding gradient problem) Auto-Encoders have also shown state-of-the-art performance in finding a representation for multiple sequential data such as video [48] and sensor data [49]. LSTM-Auto Encoders have shown to result in improvement of accuracy in audio classification as well [50] and therefore, LSTM-Auto Encoders are potential candidates for creating an efficient representation of the series of audio frames.

1.3. Research questions and contributions

1.3.1. Can knowledge transfer from image classification improve the accuracy of a One/Few Shot Learning model fine-tuned for sound classification?

Although transfer learning between different areas (image and audio) has been investigated, its effect is not yet proven for One/Few Shot Learning in audio classification. Three famous state-of-the-art models have been chosen and the effect of transfer learning has been investigated using a pre-trained model on ImageNet and fine-tuned on ESC-50 data set as the feature extractor of the Siamese Network in One/Few Shot Learning.

[6] Further explanation in 2.8

[7] Refer to 3.5.2

[8] Explained in 2.6

[9] Explained in 2.7

1.3.2. How much transfer learning from an audio classification model to the feature extractor can improve the result of the One/Few Shot Learning model?

To analyze the effect of transfer learning from an audio classification model to the feature extractor, two scenarios are investigated on the same model architecture: i) A model initialized using random weights and trained on the ESC-50 dataset from scratch and ii) A similar model initialized by transferring knowledge from an audio classification model which was trained on AudioSet and then fine-tuned on the ESC-50 dataset.

1.3.3. In Siamese Network, can a neural network learn to compare the feature extractors' outputs and improve the One/Few Shot Learning model accuracy?

In the original idea, Siamese Networks use a comparison function like cosine distance to find the probability of two feature vectors (representing two inputs) being in the same class. It is proposed to design and train a neural network to act as this comparison function. The effect of using this proposed comparison model is investigated and compared to the original comparison function (cosine distance).

1.3.4. How can the One/Few Shot Learning Model look at more than one time-frame?

Using a Convolutional Neural Network (CNN) as the first layer of feature extractor, a LSTM-Auto Encoder as the second layer of feature extractor, the Siamese Networks idea (having two identical networks as feature extractor), and a Neural Network as comparison function, a novel One/Few Shot Learning model is proposed to find features in each time frame, compress them as one whole feature vector and make the decision based on not only the data from the last time frame but also on how the sounds have changed during a longer period.

1.4. Thesis Structure

The remainder of the thesis is structured as follows:

- Chapter 2 presents a summarized description of Machine Learning terminology used in this research, followed by a brief introduction to the Dataset used in training and testing.
- Chapter 3 presents the experimental methods including a detailed description of proposed model architectures for each test.
- Chapter 4 presents the results of proposed tests showing the effect of different approaches. The research questions are all answered here.
- Chapter 5 presents a summary of the outcome, obstacles in this research, and recommendations for possible future works.

Chapter 2. Neural Networks and Deep Learning

This chapter provides a high-level description of Deep Learning with a focus on concepts used in the proposed system.

The artificial intelligence algorithms were originally designed to copy how the human brain learns from environmental activities; thus, they were called Artificial Neural Networks (ANN). The term “deep” is used to show that these networks have more layers and can grow much bigger compared to conventional systems.

2.1. Fully Connected Layers

Fully Connected Layers are consisted of a set of neurons where each neuron is connected to all the input data that comes to the layer, where the input data could be the output of a different layer or it could be the model’s input. These neurons are mathematical replicas of how a biological neuron works in human body. Figure 2-1 shows the structure of a single biological and mathematical neuron.

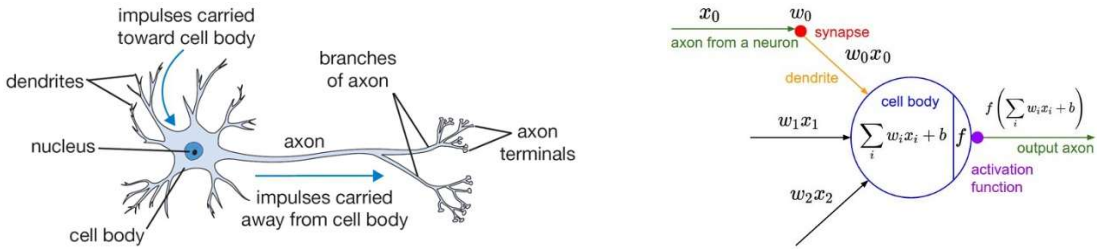

Figure 2-1: A schematic drawing of a biological neuron (left) and its mathematical model (right)[10]

From a mathematical perspective, these neurons can be split up into two different parts:

1. The body where a linear combination of the input (with a bias value) is calculated using the following function,

[10] https://www.gabormelli.com/RKB/Artificial_Neuron

$$y_i = \sum_{j=1}^{n} (w_{ij} * x_j) + b_i \quad (1)$$

where **i** is the neuron number, **j** is the input number, **w** is the weight for the input, **x** is the input and **b** is the bias.

2. The activation function which adds non-linearity to the system by applying a nonlinear function to a linear combination of the input.

2.2. Activation Functions

Activation functions are integral parts of any neural network as they allow the model to go beyond the trivial linear problems and generalize and adapt with variety of nonlinear combination of input passing through multiple layers. There are many activation functions, but the following 3 were used throughout this work which are explained as below:

2.2.1. Rectified Linear Unit (ReLU)

A piecewise function that outputs the input directly for positive inputs and returns zero otherwise. Figure 2-2 shows the plot for this function which describes as,

$$f(x) = \max(0, x) \quad (2)$$

where **x** is the input.

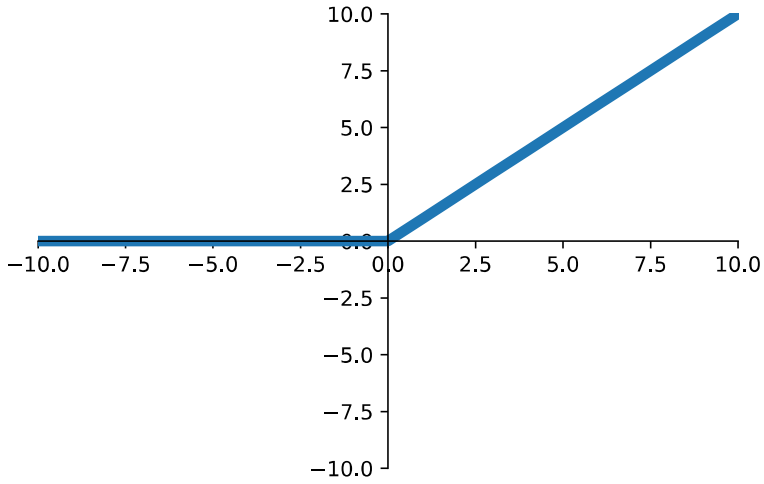

Figure 2-2: ReLU activation function

ReLU became extremely popular due to its robustness to vanishing gradient and sparsity. Another advantage of ReLU is the low computational costs compared to much more complex activation functions.

2.2.2. Sigmoid

A mathematical function that has an “S” shaped curve. A common example of such a function is the logistic function shown in Figure 2-3 and equation (3) [51]. Such function would be monotonic, continuous, and differentiable everywhere such as,

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

where **x** is the input.

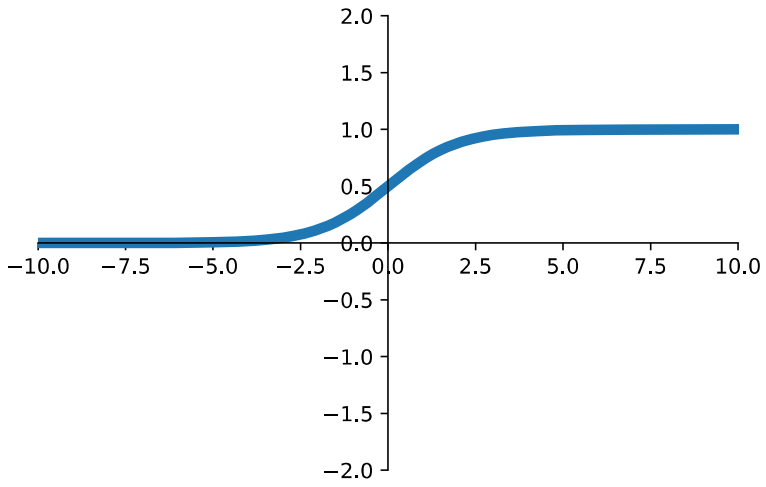

Figure 2-3: Sigmoid activation function

2.2.3. Hyperbolic Tangent (Tanh)

This function acts like a stretched and shifted version of the Sigmoid function. The output is in the range of [-1, 1] vs Sigmoid's output which is in the range of [1, 0], as shown in Figure 2-4. Tanh is calculated using,

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4}$$

where **x** is the input.

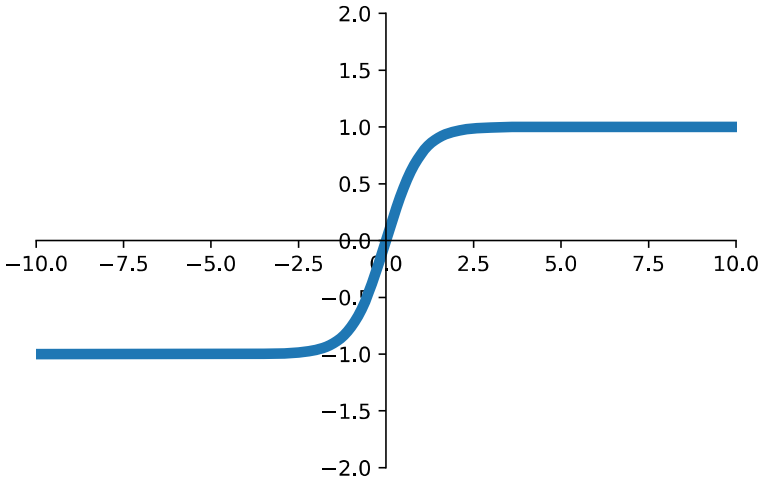

Figure 2-4: Tanh activation function

2.3. Convolutional Layer

Convolutional Neural Networks (CNNs), introduced by a postdoctoral computer science researcher in the 1980s [52], are one of the most popular Neural Networks to work with multidimensional data (like a 2D photo or a 2D representation of audio). CNNs are typically consisted of a series of convolutional layers followed by few fully connected layers. Convolutional layers are made of a set of filters (a set of parameters usually having a smaller size than the input). The filter parameters are learned through the training process. These filters will create the output of the layer by sliding across the input (going over every spatial position) and finding the sum of dot product of the filter and the input (convolution of the input and the filter). This would allow Convolutional Neural Networks to successfully catch spatial and temporal dependencies of the input while having fewer parameters using the application of relevant filters. Figure 2-5 shows how a filter is applied to a 2D input.

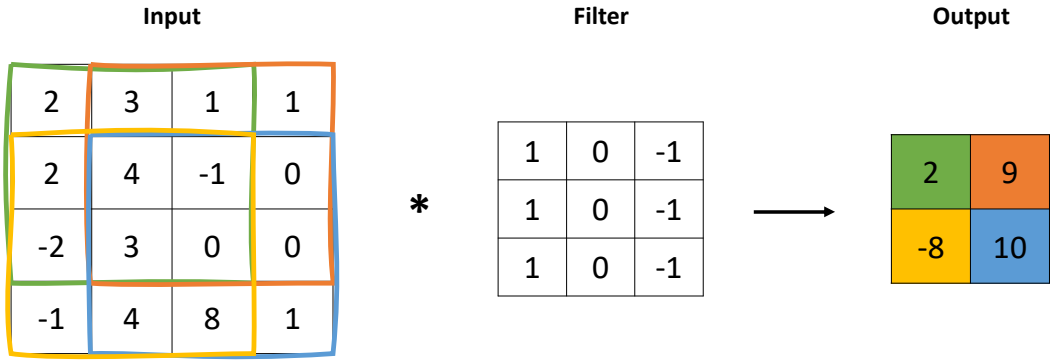

Figure 2-5: Filters in CNNs. The filter is applied on each colored box and creates a cell in the output. Colors in the output are matched with the corresponding box color. The * denotes a convolution.

CNNs have been primarily used in visual recognition contexts, but there are also many successful applications in audio-related problems such as speech recognition and large scale audio classification [53][54][55].

2.3.1. Pooling

The pooling layer is responsible for spatial size reduction. Pooling layers result in decreased computational power and they are useful for extracting the dominant features in a positional-invariant and rotational way that can maintain the process of effective training. Max and Average pooling are the common ways of implementing the pooling layer as shown in Figure 2-6. In Max pooling, the filter returns the maximum value from the input while the Average pooling returns the mean of the input.

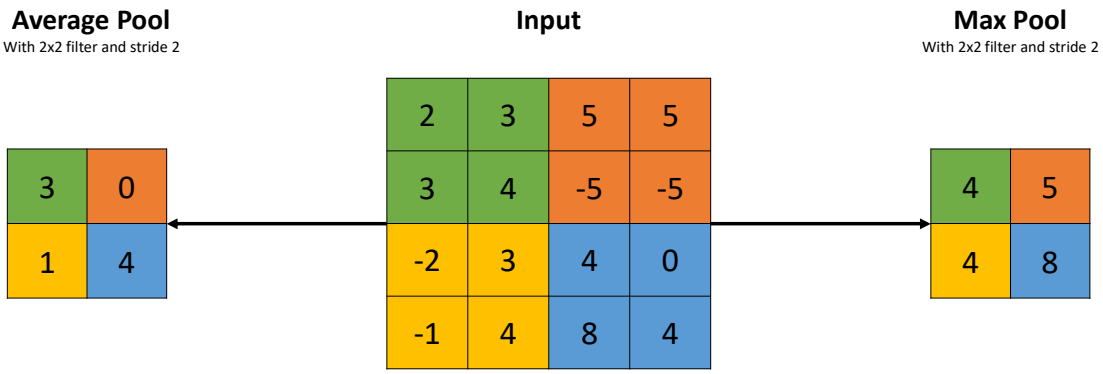

Figure 2-6: Max Pooling vs Average Pooling

2.3.2. Batch Normalization

Introduced by two Google researchers in 2015 [56], batch normalization makes the network train faster and more stable by re-centering and re-scaling of the hidden layers. The underlying logic behind batch normalization is still under investigation but the effectiveness of it has been proven [56].

2.3.3. Regularization

It is a technique used to control the process of fitting the model to the training set and avoid overfitting. This method tries to discourage the model to learn too much complex function that fits perfectly into the training set but would perform poorly on the unseen test set [57].

2.4. Siamese Networks

As discussed in the previous chapter, the Siamese Networks were initially introduced by [34] for signature verification. The idea behind Siamese Networks is to have two identical networks extract a set of features from input and a labeled sample. Then a comparison is made on the extracted features to determine the similarity of the input and the labeled sample. Figure 2-7 shows the structure of a Siamese Network.

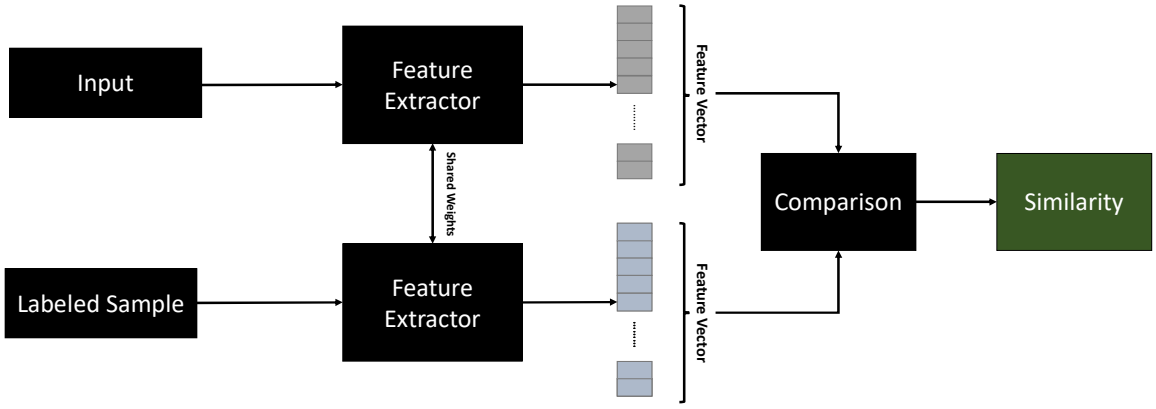

Figure 2-7: Structure of a Siamese Network

2.5. Transfer Learning

It is a machine learning method where a model trained for a task is reused as a starting point in another (similar) task. A model (or part of it) that has been trained in one setting will be exploited to improve generalization in another setting. The idea works the same as if a person knows how to ride a bike, they can learn how to ride a motorcycle much easier and even better (with the same training data) than a person with no experience at all.

2.6. Recurrent Neural Network (RNN)

A class of neural networks created to analyze sequential data and time series. Figure 2-8 shows the structure of a RNN where the model looks at the input at each time step and a variable, called state, from itself in the previous time step. This allows RNN to make its prediction not only based the input, but also based on the inputs in the previous time steps, which makes RNN a perfect fit for analysing sequential data and time series since it can make predictions considering the history of the data. With larger sequences, because the state of the model is fed back into the model at every time step, vanishing/exploding gradients become a common problem happening during the training. Vanishing gradient is when layers more distant from the output would not change much during the training (compared to the last layers of the model) and exploding gradient is when there is an exponential growth in the model parameters.

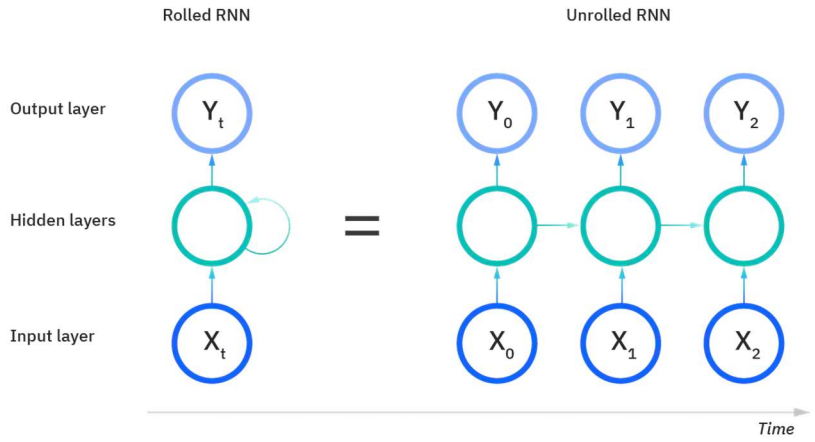

Figure 2-8: Structure of a Recurrent Neural Network[11] with x_t being the input and y_t being the output at time t. The model looks at the input at each time step and also the state of itself in previous time step. Rolled RNN represents the whole neural network and the unrolled RNN represents the model unrolled over a input sequence sample (with size: 3) and shows how the model is passing the state from one time step to the next one (green arrows).

2.7. Long Short-Term Memory (LSTM)

As the time window of the input gets larger, the problem of vanishing or exploding gradient becomes more common in a Recurrent Neural Network. LSTM is an architecture specially designed to address such problems [58]. The architecture of each memory cell guarantees constant error flow within its Constant Error Carousell (CEC). Figure 2-9 shows how an LSTM cell controls the information flow using three gates. The forget gate can control how much of the previous cell state passes on which can guarantee a Constant Error Carousell.

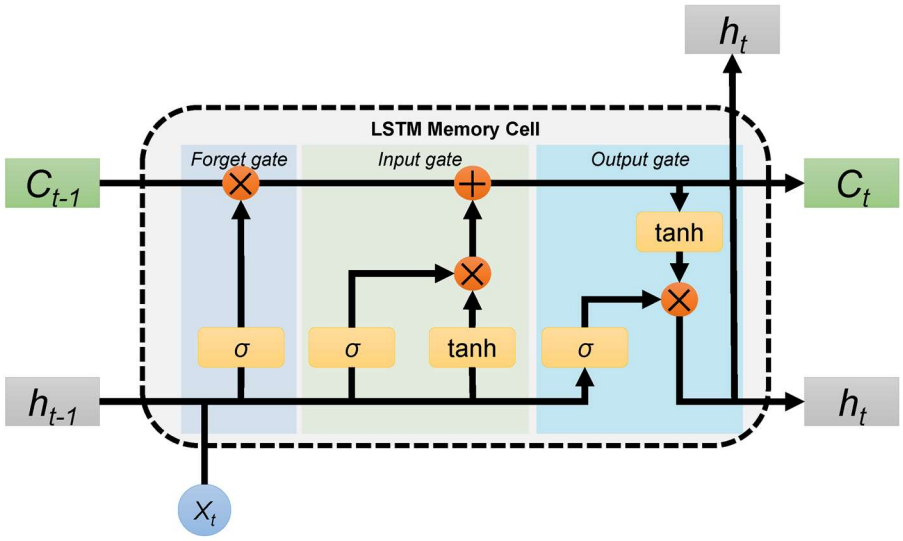

Figure 2-9: Structure of LSTM Cells [59] **with C being the Cell State at each time step, X being the input at each time step and h being the output of the model at each time step. Forget gate controls how much of the cell state passes on to the next time step.**

2.8. Auto Encoders

Initially introduced in 1968 to find an internal representation of input by error propagation [46], an auto encoder can be considered as an unsupervised technique with one goal , “representation learning” or learning to create a vector for each input that perfectly represents the input. It can be used to reconstruct the input with minimum loss. As shown in Figure 2-10, it is a neural network that is trained to reconstruct its input while having a bottleneck – a layer that is limited by size where not all parts of the input can pass through. Since the model must reconstruct the input after this bottleneck and the size of this bottleneck is limited, it tries to create a vector that explains the input in the bottleneck which can be interpreted as the “representation” of the input. Figure 2-10 shows how an image is converted to a vector (output of the bottleneck layer) and then reconstructed. The reconstructed image is not exactly the same as the input but the better the model learns to create the presentation the less loss there will be.

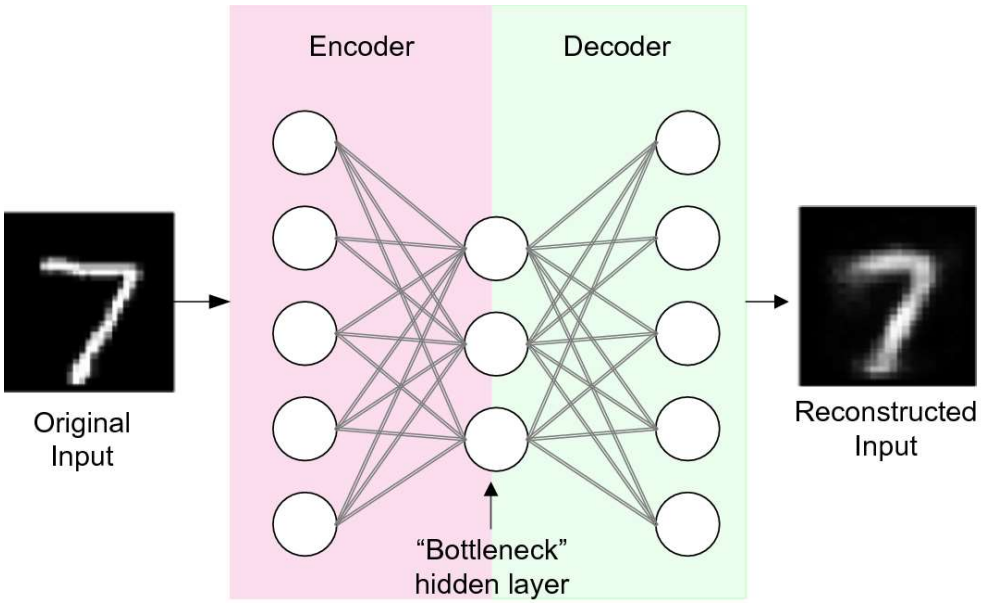

Figure 2-10: A simple Auto Encoder[12], The encoder creates a representation of the input at the bottleneck and the decoder usese the representation to reconstructe the input.

2.9. Datasets

2.9.1. Dataset for Environmental Sound Classification (ESC-50)[60]

This is a data set of 2000 5-sec audio samples from 50 different classes as listed in Table 1:

Table 1: Ontology of the ESC-50 Dataset

Animals	Natural soundscapes & water sounds	Human, non-speech sounds	Interior/domestic sounds	Exterior/urban noises
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door, wood creaks	Car horn
Frog	Chirping birds	Coughing	Can opening	Engine
Cat	Water drops	Footsteps	Washing machine	Train
Hen	Wind	Laughing	Vacuum cleaner	Church bells
Insects (flying)	Pouring water	Brushing teeth	Clock alarm	Airplane
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking, sipping	Glass breaking	Hand saw

[12] https://medium.com/analytics-vidhya/what-is-auto-encoder-in-deep-learning-5d668f94651b

ESC10 is a subset of ESC50 that includes only these 10 classes:

Dog, Rooster, Rain, Sea waves, Cracking fire, Crying baby, Sneezing, Chainsaw, Helicopter, Clock tick

The audios are all 5 seconds long and each class has exactly 40 samples. The clips were manually extracted from the Freesound[13] project.

This would be a perfect dataset to work on for One/Few Shot Learning because of the following reasons:

1. **The diversity of classes:** In training process of any classification model, it is common to keep a part of the training data for testing and validation. In conventional classification models, this is done by keeping data from each of the classes while in One/Few Shot Learning one or more classes must be entirely reserved for testing (unseen by the training process). This makes the diverse list of classes in ESC-50 a perfect candidate for the job as we can easily exclude 10 or even more classes from the training process and still have 40 classes to train the model with. Also, One/Few Shot Learning model is designed to generalize and learn to find a general answer for a wide variety of possible classes. This means having a lot of samples for few classes can create a biased model toward those specific classes while a diverse dataset with diverse list of classes can help the model to better generalize, as it is supposed to.

2. **Handpicked Clips:** Audios in this dataset are handpicked and hence do not need a cleaning process. This guarantees that the training process does not get affected by imperfections of the dataset.

3. **Balanced Dataset:** Each class has exactly 40 samples with same length which would not cause an overfitting problem like an unbalanced dataset.

[13] freesound.org

Chapter 3. Sound Classification with One/Few Shot Learning

A summary of methods explored in this thesis is shown in Figure 3-1. This chapter gives an explanation on each part starting with the major parts of the One/Few Shot learning with Siamese Networks followed by proposed adjustments to improve the accuracy of the model.

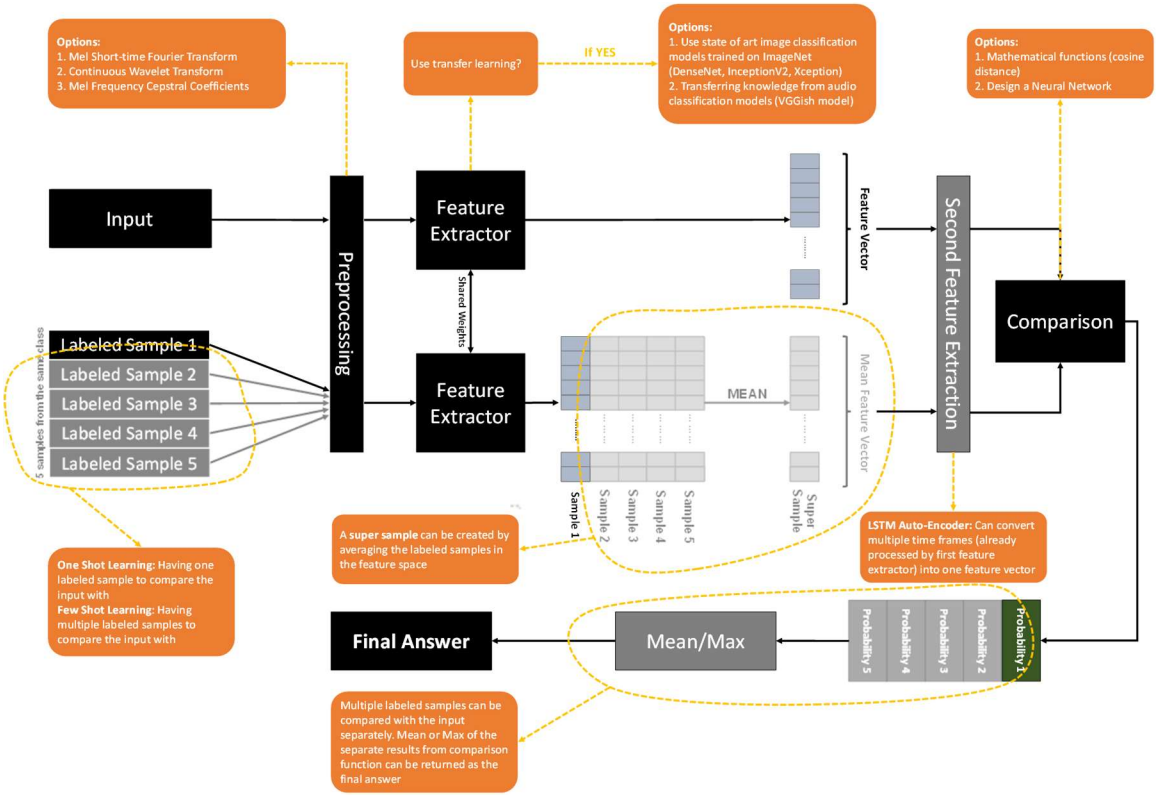

Figure 3-1: Summary of methods that were explored to improve the accuracy of a One/Few Shot Learning model with Siamese Networks. The orange boxes give a brief explanation on each component. The parts in grey are optional and both scenarios with and without them are investigated in this thesis.

The major parts of a One/Few Shot Learning model with Siamese Networks can be broken down into three parts as shown in Figure 3-2. Following questions were addressed regarding these major parts:

1. How to process the audio so that the model can extract the most possible information – **preprocessing.**

2. How to create a model that can extract the features of the processed audio – **feature extraction**.

3. How to compare the features extracted from the input with the features extracted from the labeled sample(s) – **comparison network**.

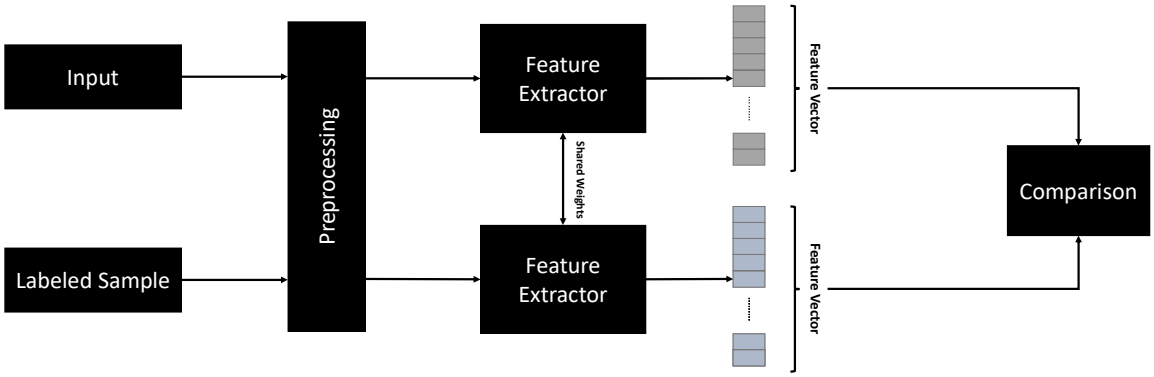

Figure 3-2: Different parts of a One Shot Learning model with Siamese Networks, input and the labeled sample go through three different stages: Preprocessing, Feature Extraction and Comparison

3.1. Preprocessing

Successful application of convolutional neural networks in audio classification and speech recognition became the motivation to search for better audio representation to achieve better training. Time-frequency representations of sounds have always played a huge role in audio signal processing. Among the common approaches, Mel Spectrogram (Short-Time Fourier Transform or STFT) has shown a great boost for convolutional neural network audio-related training tasks [61]. Most of the proposed models were created based on Mel Spectrogram but Continuous Wavelet Transform (CWT) and Mel Frequency Cepstral Coefficients (MFCC) [62] were also used alongside the Mel Spectrogram to create a 3-layer (same as RGB in pictures) input when investigating transfer learning from image classification models.

3.1.1. Mel Short-time Fourier Transform

The Fourier Transform is one of the most powerful analytical tools in many different scientific applications. Waveform data can be transformed from the time domain – i.e., amplitudes of a measurement (air pressure for audio) in each time step, into the frequency domain using Fourier Transform. The transformation is done by breaking a

waveform down to a series of sinusoidal terms with unique magnitude, frequency, and phase. An efficient way of implementing Fourier Transform on Discrete data such as audio is to use Fast Fourier Transform (FFT), an implementation of Discrete Fourier Transform (DFT) with almost the same result and significantly less computation time [63].

Fourier Transform makes the analysis of a signal's frequency contents easier. However, in environmental sound classification and many other similar tasks (non-periodic signals), the frequencies change over time and these changes sometimes show the characteristic of the sound. **Short-Time Fourier Transform** (STFT) is performing FFT on multiple windowed segments of a signal (usually with overlap). The result of an STFT is called a **Spectrogram**, a series of FFTs stacked next to each other showing the power of each frequency in different time steps. Figure 3-3 shows spectrogram of a dog barking sound compared to the sound plot.

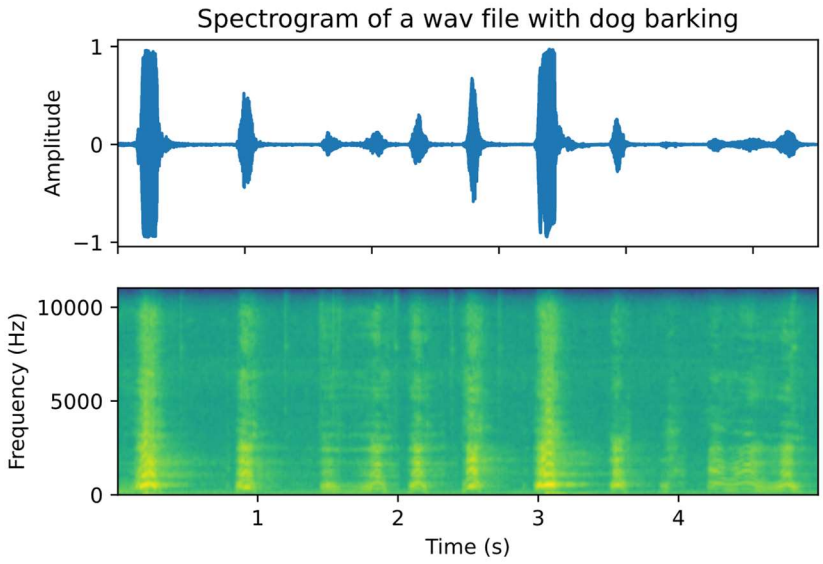

Figure 3-3: Spectrogram of a dog barking sound. Barking creates a sound with higher energy than background noise and can be detected where higher amplitude is seen. In frequency domain, lighter colors indicate higher energy in that specific frequency bin at that time.

Human ears do not perceive sound frequencies on a linear scale. The mel scale was introduced to scale sound frequencies in a way that pitches are judged at the same distance as much as they are distanced to a human listener. A common practice for

converting frequencies (f) into a scale that is similar to how a human interpret frequency (m) is,

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{5}$$

Using equation (5), the same Spectrogram in Figure 3-3 can be converted to mel-scale. The converted mel scale Spectrogram is shown in Figure 3-4.

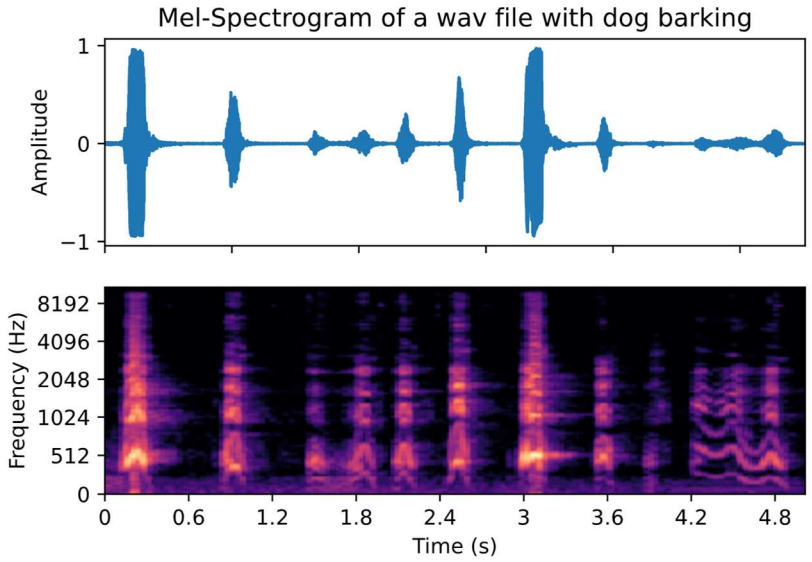

Figure 3-4: Mel-Spectrogram of a dog barking sound. In frequency domain, lighter colors indicate higher energy in that specific frequency bin at that time.

3.1.2. Continuous Wavelet Transform

The Continuous Wavelet Transform compares the signal to the shifted and compressed or stretched version of the wavelet. Here the inner product is used to find the similarity of a selected wavelet (a localized wave) and the input. The similarity of the input with selected wavelet is computed as,

$$C\big(a, b, f(t), \psi(t)\big) = \int_{-\infty}^{+\infty} f(t) \frac{1}{a} \psi^*\left(\frac{t-b}{a}\right) dt \tag{6}$$

where $*$ denotes the complex conjugate, **a** is the scale parameter, **b** is the position parameter, **ψ** is the wavelet, **t** is time, and **f(t)** is the input signal.

By definition, any wavelet can be used in equation (6). For this research, first-order derivative of the Gaussian wavelet, as shown in equation (7), is used.

$$\psi(t) = C * e^{-t^2} \tag{7}$$

where **t** is time and **C** is a normalization constant.

A Continuous Wavelet Transform is done on the same dog barking sound used in Figure 3-3 and Figure 3-4 and the result is shown in Figure 3-5. This figure shows the similarity of the audio with the Gaussian wavelet over time. Lighter colors indicate a higher similarity if the input at that time with the Gaussian wavelet.

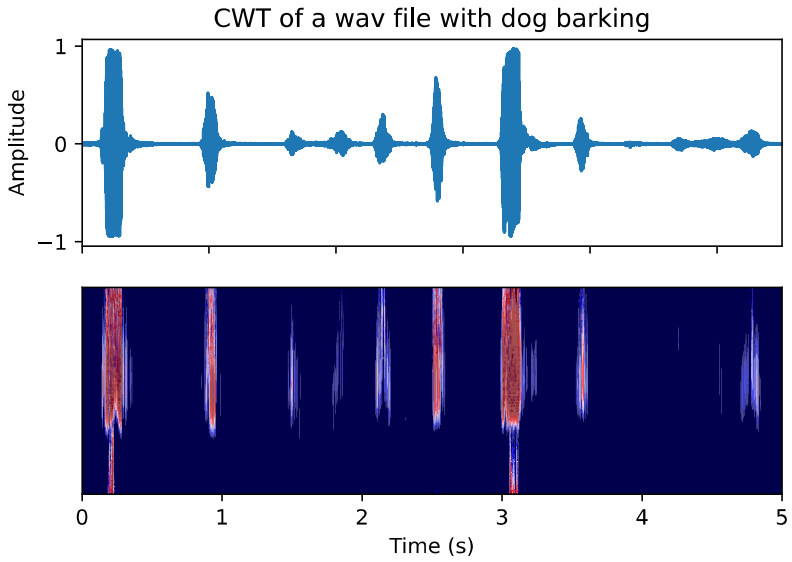

Figure 3-5: CWT of a signal with Gaussian wavelet, lighter colors indicate higher similarity.

3.1.3. Mel Frequency Cepstral Coefficients(MFCC)

MFCC, is obtained by applying Fourier Transform on log of magnitudes of a Fourier Transform in Mel Scale. The process of finding MFCCs is shown in Figure 3-6. Since in MFCC a spectrum of a spectrum is calculated, the result is neither in the time domain nor the frequency domain and hence it was named the quefrency domain [64].

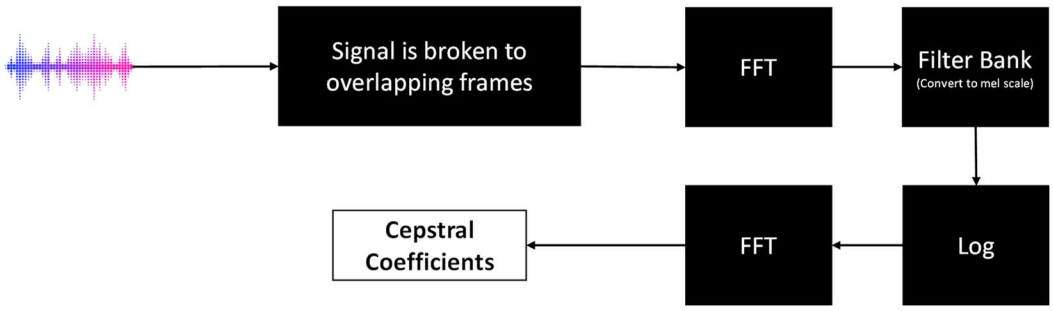

Figure 3-6: Steps in calculating Mel Frequency Cepstral Coefficients

Applying the steps described in Figure 3-6 on same dog barking sound as before would result in MFFCC Coefficients shown in Figure 3-7.

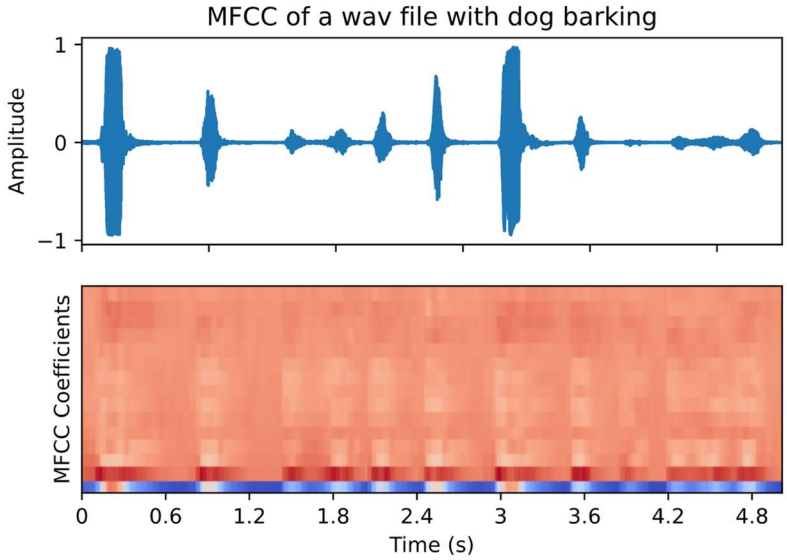

Figure 3-7: MFCC of a dog barking sound, absolute value of coefficients increases as the color changes from red to blue.

3.2. Feature Extraction

A common feature extraction design strategy for One/Few-Shot Learning models is to use part of a network used for classification of a similar task. In the case of environmental sound classification, convolutional neural networks have proven to be a wise choice [65][66][67]. Convolutional neural networks have also proven to achieve astonishing results on very limited computing resources for edge devices (devices with

limited computational and potentially small amount of power such as a cell-phone) [68]. A variation of convolutional neural networks was tested in the following steps to find the optimum feature extraction model starting with image classification models.

3.2.1. State of Art Image Classification Models

There might be a significant difference between an image and a sound representation like the ones discussed in preprocessing stage, but studies have shown that models trained for image classification (e.g., on ImageNet [26]) can be a great baseline network for sound classification tasks [35]. Considering how auditory and visual brain regions in a human brain have shared representational structure [69], transferring knowledge between these two domains should be considered a possible way of achieving better performance. ImageNet has become a benchmark for image classification models, so the following state of art models with outstanding performance on ImageNet were used to explore the effect of transferring knowledge from an image classification task to a One/Few-Shot Learning problem (models with high computational costs like NASNetLarge [70] were not considered despite their high performance):

DenseNet201 [71]***::***

A shorter connection between the layers close to output and layers close to the input can make a model more accurate and efficient as shown by recent studies [71]. Dense Blocks connect every layer to all preceding layers to create short connections between layers close to the output and the ones close to input. Figure 3-8 shows a network with three Dense Blocks where layers within a block are connected to every other layer in a feedforward fashion.

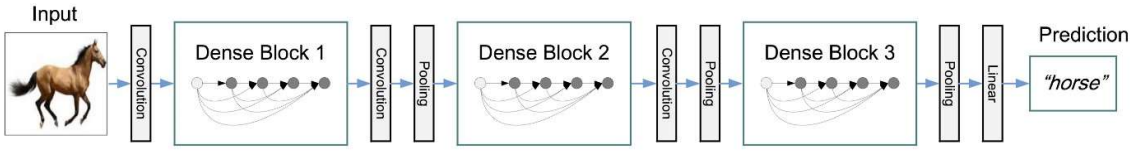

Figure 3-8: A deep DenseNet with 3 Dense Blocks [71]**, each layer whitin a dense block is connected to all other layers in that block in a feedforward fashion.**

DensNet201 is a network with 201 layers in total containing four Dense Blocks. Table 2 shows a summary of DesneNet201 architecture.

Table 2: A summary of DenseNet201 architecture

Layer	Description
Convolution	7 × 7 conv, stride 2
Pooling	3 × 3 max pool, stride 2
Dense Block (1)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix}$ × 6
Transition Layer (1)	1 × 1 conv
	2 × 2 average pool, stride 2
Dense Block (2)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix}$ × 12
Transition Layer (2)	1 × 1 conv
	2 × 2 average pool, stride 2
Dense Block (3)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix}$ × 48
Transition Layer (3)	1 × 1 conv
	2 × 2 average pool, stride 2
Dense Block (4)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix}$ × 32
Classification Layer	7 × 7 global average pool
	1000D fully connected, Softmax

InceptionV3 [72]

As a variant of GoogleNet [73], InceptionV3 focuses on computational costs by modifying previous architectures and has proven to be more computationally efficient compared to GoogleNet[72]. A summary of InceptionV3's architecture is shown in Table 3 based on the original paper.

Table 3: A summary of InceptionV3 architecture

Layer	Patch Size/Stride	Input Size
Convolution	3×3/2	299×299×3
Convolution	3×3/1	149×149×32
Convolution (Padded)	3×3/1	147×147×32
Pooling	3×3/2	147×147×64

Convolution	3×3/1	73×73×64
Convolution	3×3/2	71×71×80
Convolution	3×3/1	35×35×192
3 × Inception	Figure 3-9: Original Inception module [73]	35×35×288
5 × Inception	Figure 3-9: Original Inception module [73]	17×17×768
2 × Inception	Figure 3-9: Original Inception module [73]	8×8×1280
Pooling	8 × 8	8 × 8 × 2048
Linear	logits	1 × 1 × 2048
Softmax	classifier	1 × 1 × 1000

Inception module, as shown in Figure 3-9, runs multiple operations (pooling, convolution) with different filter sizes (1x1, 3x3, 5x5) in parallel to have the result of different options (different combination of operations and different filter sizes) with one single module.

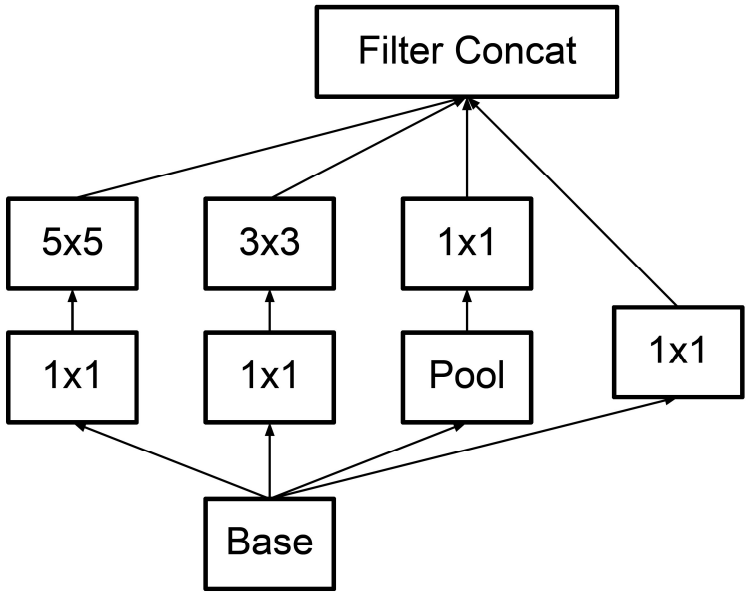

Figure 3-9: Original Inception module [73]**, a combination of different operations and filter sizes which are calculated in parallel**

In InceptionV3 the fully connected layer of the auxiliary classifier is normalized as well as the convolutions.

Xception [74]

Xception is called the extreme version of the Inception model. By changing the order of operation and presence/absence of non-linearity (activation functions), the model has a slightly better performance than InceptionV3. Residual shortcuts, as originally proposed by ResNet [75], are placed for all flows. Figure 3-10 shows a summary of Xception model architecture.

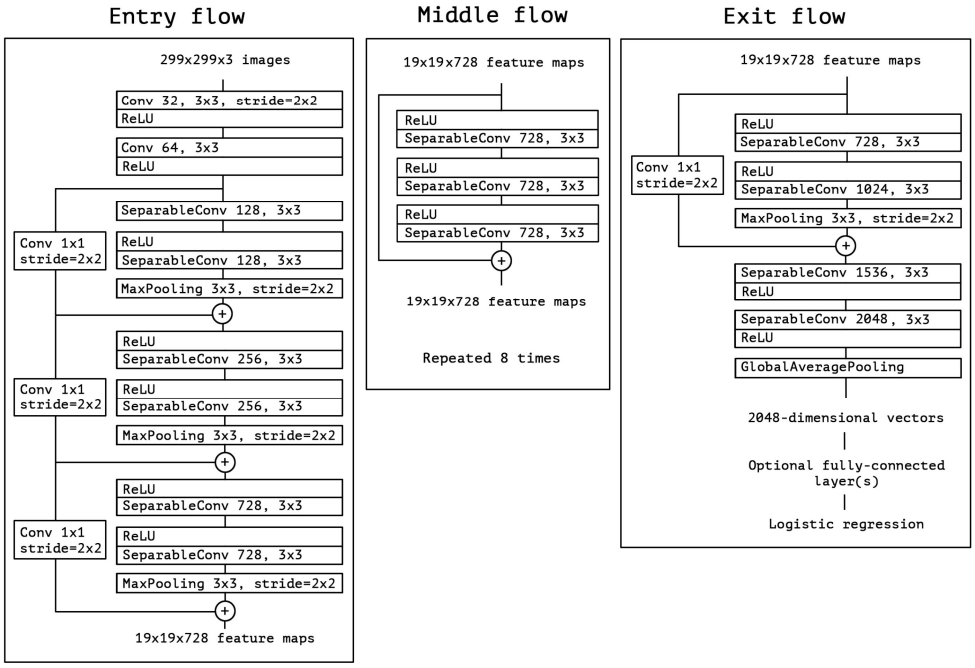

Figure 3-10: The Xception Model architecture splited into three parts which are connected in order of Entry, Middle and Exit flow. Residual shortcuts are placed in each flow on the left side.

Summary of the accuracies of the selected Image Classification Models

ImageNet validation is among one of the most popular benchmarks for image classification models. All these three chosen models were tested on ImageNet validation dataset (by the original authors) and reported a Top-1 and Top-5 accuracies where a Top-**N** accuracy means whether the model chose the right answer as one of the first **N** most probable classes for the input.

Since these models were all pre-trained on ImageNet (which contains images with varieties of sizes) a fixed input shape of [3, 299, 299] was chosen when training the models on ImageNet and all the images where resized or cropped to fit this input size.

This input size indicates that when using this architecture for One/Few Shot Learning audio classification, each audio should be converted into 3 layers each with the size of 299x299. For the layers, Mel Short-time Fourier Transform (Mel-Spectrogram), Continuous Wavelet Transform (CWT), and Mel Frequency Cepstral Coefficients (MFCC) are used to create the required input.

All the mentioned models have a Softmax layer as the last layer to classify the input. By removing the last Softmax later, the new last layer would have the output shape of 2048 which can then be interpreted as a **feature vector** of the input. Table 4 shows a summary of the three chosen classes, all the models have state-of-the-art performances and based of their reported accuracies, Xception has proven to be the most optimal model (almost same size, higher accuracy).

Table 4: A summary of chosen image classification models on ImageNet validation dataset.

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
DenseNet201	80	77.3%	93.6%	20.2M	402
InceptionV3	92	77.9%	93.7%	23.9M	189
Xception	88	79.0%	94.5%	22.9M	81

3.2.2. Transferring knowledge from Audio Classification models

AudioSet [38] is one of the largest audio datasets available online. It was made based on a large-scale video data set now known as YouTube-8M [76]. With the initial release of AudioSet, a 128-dimensional embedding was provided for each segment (a 0.96s moving window) of the audio. These embeddings were produced using a VGG-like [77] classification model. The weights from the classification model trained for classifying AudioSet were used as a starting point for the VGGish feature extractor shown in Figure 3-11. The input for the VGGish model would be a Mel-Spectrogram of the audio created using the configuration in Table 5 and creates a 128-dimensional feature vector which will be fed eventually to the comparison network as shown in Figure 3-2.

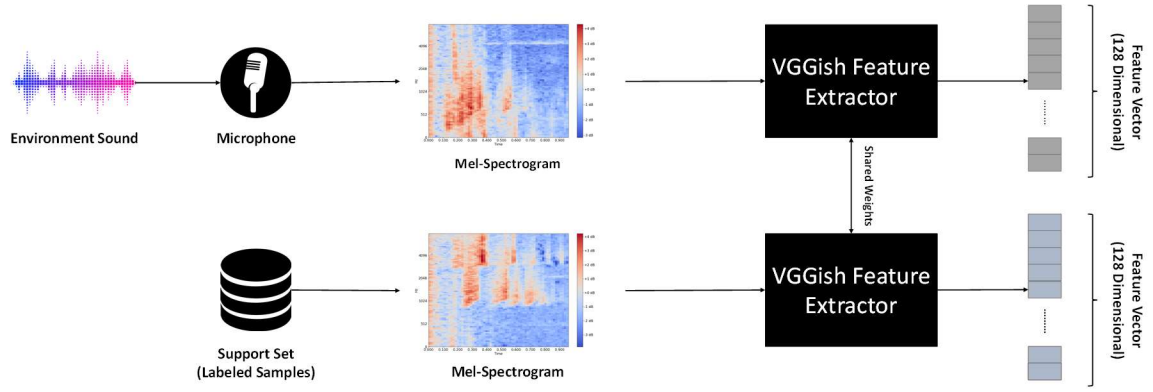

Figure 3-11: A representation of how the VGGish feature extractor would fit in the Siamese Network

The preprocessing required for VGGish feature extractor is calculating a Mel Spectrogram of the audio with the configuration shown in Table 5.

Table 5: VGGish model preprocessing configuration

Number of Frames	96
Number of Bands	64
Sample Rate	16000
STFT Window Length	0.025s
STFT Hop Length	0.01s
Window Size	0.96s

The VGGish model is originally derived from a popular image classification model called VGG [78]. As shown in Figure 3-12, the modification was done to create the 128 feature vector from the output of the model, replacing the Softmax layer used for classifying images in the VGG model.

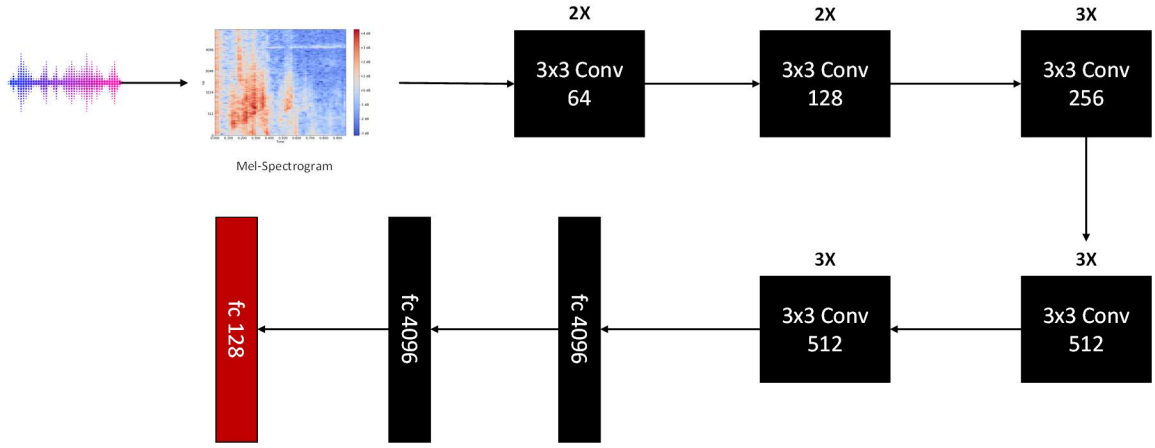

Figure 3-12: A VGGish network is derived from VGG-16 (16 layer VGG Model [77] **originaly designed for image classification). Black cells are the same as VGG and the last red fully connected cell is to create a 128-dimensional feature vector.**

3.3. Comparison Network

After finding a feature vector that can represent each audio, a comparison must be made between the feature vector of input and the feature vector of the labeled sample. Cosine similarity is a popular approach to compare two feature vectors for One/Few Shot Learning models. In addition to testing the cosine similarity as comparison network, a Neural Network is also proposed to do the comparison. To train such Neural Network, the goal of the training should be defined in a way that eventually the result would show the probability of the input belonging to the same class as the labeled sample. Cosine similarity and the proposed Neural Network as the comparison network are further discussed in this section.

3.3.1. Cosine Similarity

One of the most common mathematical approaches for comparing feature vectors is cosine similarity. It would result in **1** for the identical feature vectors, **0** for orthogonal vectors, and **-1** for exact opposite vectors which can be transformed to a probability range of [0, 1] for being in the same class. The probability of input and the labeled sample being in the same class is calculated for feature vectors A (input) and B (labeled sample) using,

$$Propbability\ of\ being\ in\ the\ same\ class = 0.5 * (Cosine\ Similarity + 1)$$

$$= 0.5 * (\cos\theta + 1) = 0.5 * (\frac{\sum_{i=0}^{n} A_i B_i}{\sqrt[2]{\sum A_i^2}\ \sqrt[2]{\sum B_i^2}} + 1) \quad (8)$$

where **A** is the feature vector of the input audio, **B** is the feature vector of the labeled sample and $\boldsymbol{A_i}$ and $\boldsymbol{B_i}$ indicate the $\boldsymbol{i^{th}}$ element in each feature vector.

3.3.2. A Neural Network to compare the results.

A model can be trained to find the probability of two feature vectors belonging to the same class. An L2 distance (squared distance of each feature) was calculated on the feature vectors and the result was fed to the proposed neural network shown in Figure 3-13 with three dense layers and a sigmoid layer at the end to limit the output between 0 and 1. This model should either be trained connected to the feature extractor as a whole or with the result of a fully trained feature extractor.

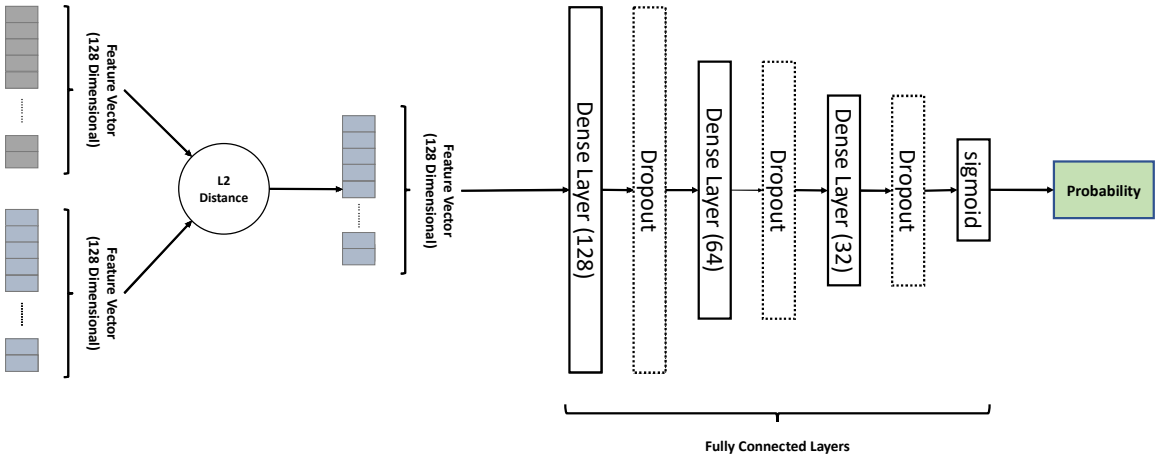

Figure 3-13: Structure of the proposed comparison Neural Network model. The L2 distance of the two feature vectors is fed to the network with three dense layers (with size of 128, 64, 32) and a sigmoid layer is used at the end to limit the outcome to the range of 0 and 1. Dropouts were used to stop the model from overfitting.

3.4. Using five labeled samples (Few Shot) instead of one (One Shot)

With more samples for a class, the system would have a better chance to find the significant features of each class. To be able to look at more than one labeled sample the two following approaches were tested:

3.4.1. Compare the results separately for each labeled sample

By running a Siamese One-Shot Learning model for each labeled sample versus the input, the probability of that input being in the same class as the labeled sample would be calculated. By having more than one labeled sample for one class, a set of probabilities were achieved for each class. The classification was done by looking at the **max** value or **mean** value of the calculated probabilities for each class as shown in Figure 3-14 where an input is converted to its feature vector alongside five labeled samples (all from the same class). Using the comparison network, the feature vector representing the input is compared to each labeled sample's feature vector creating five separate similarity scores (one for each pair). The mean and max value of the five similarity scores represents the probability of the Input being from the same class as the five labeled samples.

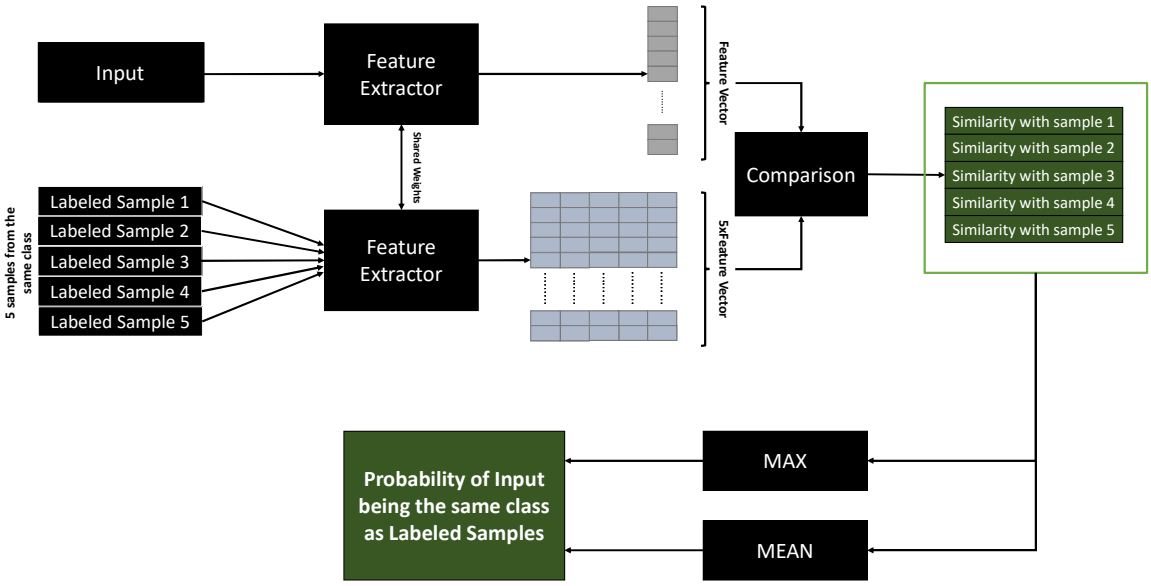

Figure 3-14: Using Few Samples (Few-Shot) instead of one to find MEAN or MAX probability of Input being in the same class as labeled samples

3.4.2. Creating a Feature Vector representing all the Labeled Samples

Using the feature extractor, each labeled sample is transformed into a 128-dimensional feature vector. A **super sample** can be created by calculating the mean of each of the 128 features over all available samples. This super sample represents the class that the input is being compared to. Figure 3-15 shows how five labeled samples (from the same class) are converted to five feature vectors representing them, then a super sample is created by taking the mean of these five feature vectors and the comparison network compares the feature vector representing the input with the super sample to find the probability of the input being in the same class as the labeled samples.

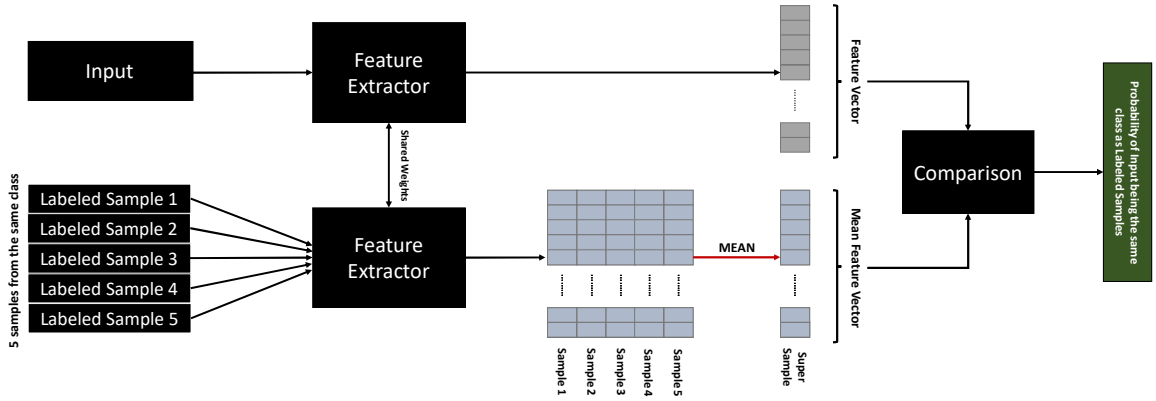

Figure 3-15: Creating a super sample from multiple labeled samples and comparing the super sample with the Input to find the probability of the Input being in the same class as the labeled samples.

3.5. Results enhancement by looking at multiple frames

In real life, humans do process the sounds they hear not only based on what is going on at the moment but also on the data they gathered before. Classifying audio by looking at more than one frame at a time can be considered a similar approach to sound classification. Each of these frames can get a probability of being in the same class as the labeled sample (mean and max of the probabilities) or the frames can be looked at all at once and a probability could be defined for the whole in one step (2-layer feature extraction).

3.5.1. Mathematical approaches: Mean and Max

By splitting a sound of 5 seconds into 0.96s frames and $\frac{2}{3} * 0.96$ second hop length, 7 different frames were created. Comparing each one with the labeled sample resulted in 7 different probabilities each showing the probability of that specific frame having the same class as the labeled input. Figure 3-16 shows how an audio with the length of 5 second is compared to a labeled sample. This results in 7 different probabilities each corresponding to a time frame in the input audio, and to achieve a single probability for the whole audio belonging to the same class as the labeled sample two approaches where tested: **Max** of these 7 probabilities or **Mean** of them as the single probability of the whole audio belonging to the same class as the input.

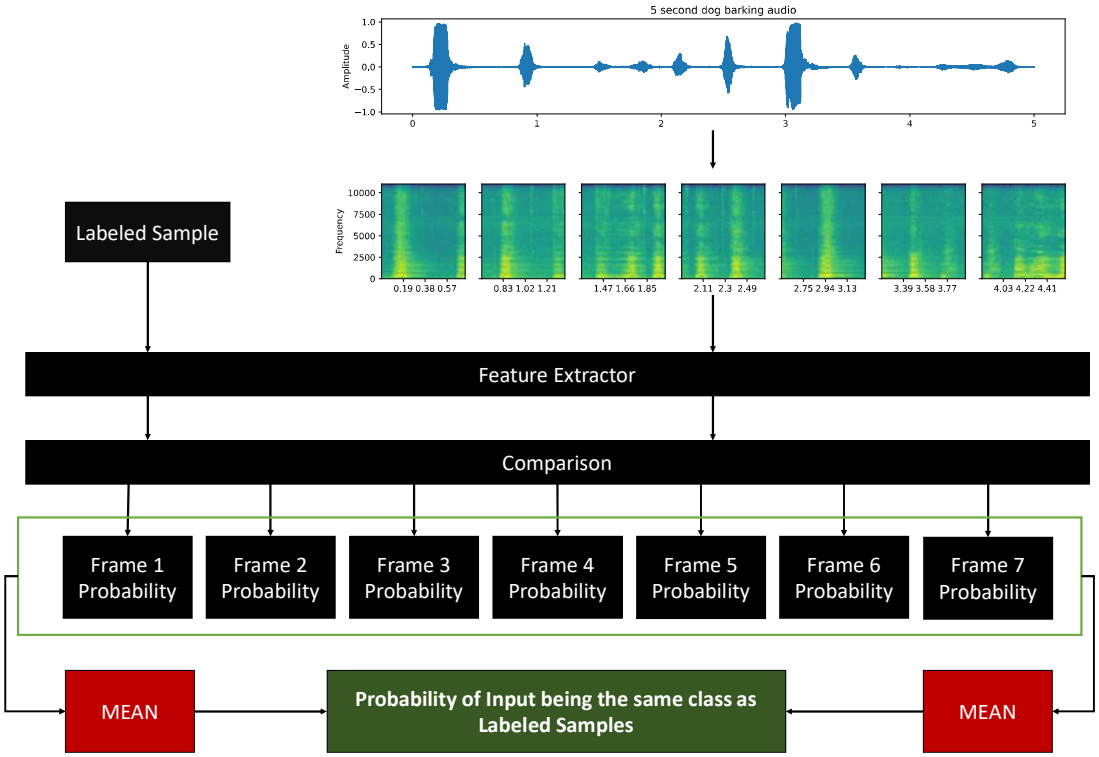

Figure 3-16: Converting a 5s audio into 7 frames of 0.96 with 2*0.96/3s hop length and finding the probability of each frame being in the same class as the labeled sample.

3.5.2. Converting multiple frames into a new feature vector (two-layer feature extraction): LSTM Auto Encoders

After training the feature extractor, using a specific type of Auto Encoders called LSTM Auto Encoder, a new embedding was created for the input and the labeled sample. Figure 3-17 shows a 5 second sound converted into 7 frames (0.96s frames and $\frac{2}{3} * 0.96$ second hop length) and a feature vector is calculated for each frame which creates 7 feature vectors each with the size of 128 (7x128 in total). A proposed LSTM Auto-Encoder as shown in Figure 3-18 converts the 7x128 feature vector representing the whole 5s audio into a new feature vector with the size of 512.

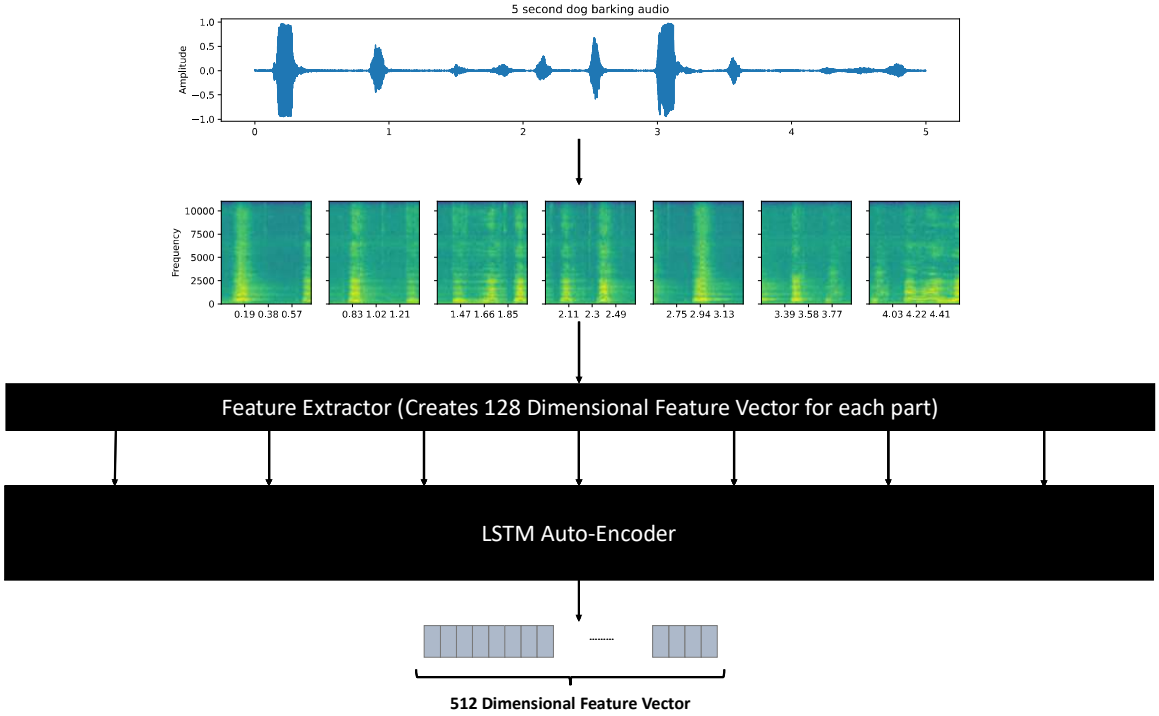

Figure 3-17: Conversion of a 5s audio into a 512 dimensional feature vector using the trained feature extractor and an LSTM Auto-Encoder

This method resulted in a feature vector that not only has the features from each frame of the input audio but also would contain info on how the sound has changed during a longer period.

The proposed LSTM Auto-Encoder, as shown in Figure 3-18, is part of a bigger model which converts the input into an embedding and then reconstructs the input using

the created embedding. If the model learns to reconstruct the input with a low error, the created embedding by the encoder in this process would contain main features of the input (because decoder only uses the features in this embedding to reconstruct the input) and can be used as the feature vector. To obtain the proposed LSTM Auto-Encoder in Figure 3-17, the connected LSTM Auto-Encoder and Decoder, shown in Figure 3-18, was trained on training dataset and a low error was achieved. It is a common practice to choose the size of the embedding vector around 2/3 of the size of the input. As the input size is 7x128, three different sizes were tested – 3x128, 4x128, 5x128. Since having smaller feature vector would result in smaller comparison function and hence lower latency, lowest size with neglectable performance loss was chosen, 4x128.

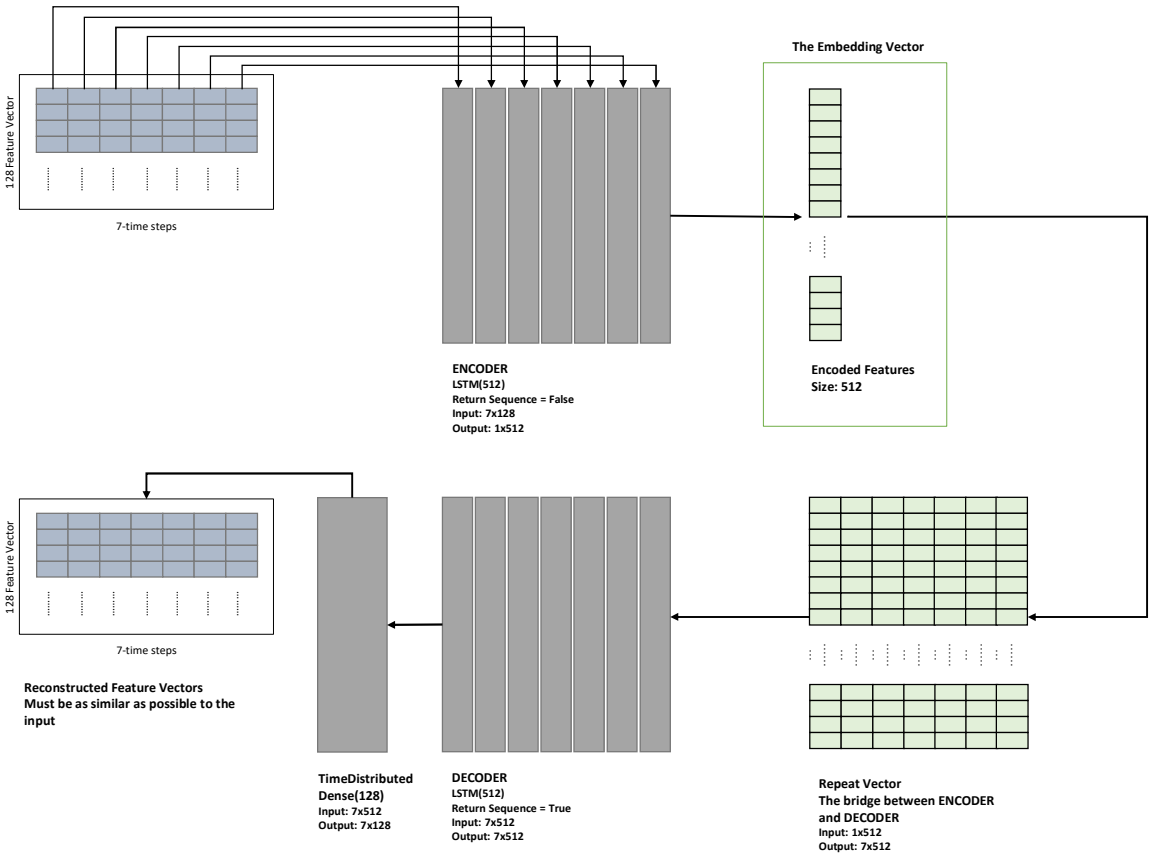

Figure 3-18: Proposed LSTM Auto-Encoder Architecture

Chapter 4 reports the test results of the methods and proposed models discussed in Chapter 3 on the Dataset for Environmental Sound Classification (ESC-50)[60].

Chapter 4. Audio Classification using One/Few Shot Learning results

4.1. Transfer Learning from Image Classification Models

Three state-of-the-art image classification models were used for investigating the effect of using models pre-trained on ImageNet (transfer learning) vs training the same model from scratch for feature extraction. The ESC-10 (10 classes) subset of the dataset is used as a validation set and the rest (40 classes) were used to (i) train the model from scratch or (ii) fine-tune the pre-trained model in ImageNet. Since the One/Few Shot Learning model is designed to compare a pair of inputs and is not initially designed for classification, the training accuracy is a pairwise accuracy and not a classification accuracy.

4.1.1. DenseNet201

Although the model can get overfitted on the training set in both scenarios (training the model from scratch or using transfer learning), as shown in Figure 4-1, the results on the validation set are higher from the beginning when using the weights from the model trained on ImageNet. This result shows that transferring knowledge from a different space can improve the accuracy and result in a more general trained model.

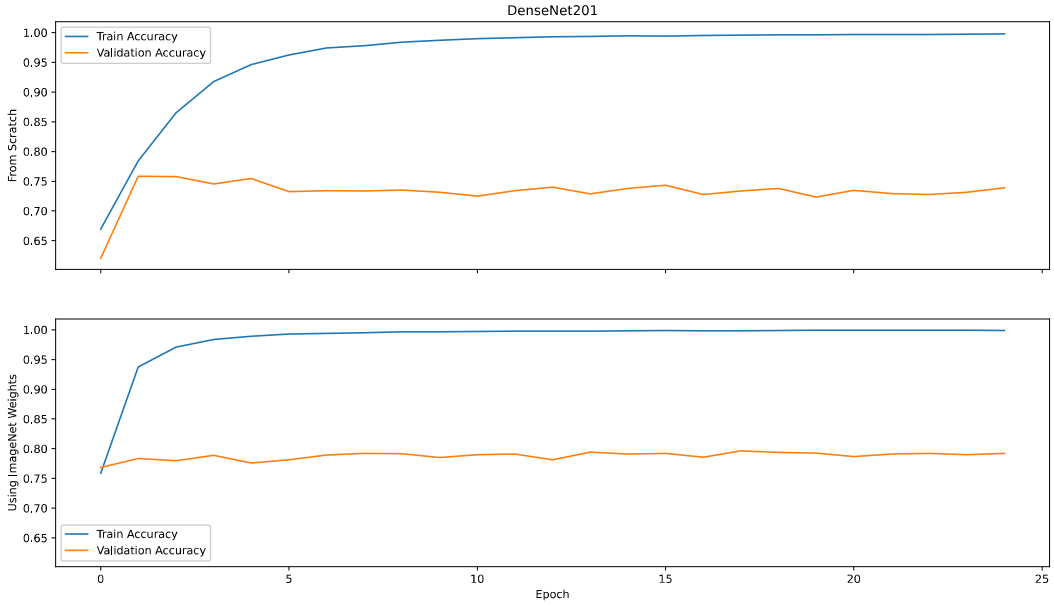

Figure 4-1: Training DenseNet201 model from scratch (Highest validation accuracy: 75.8%) vs using pre-trained weights and fine-tuning the model (Highest validation accuracy: 79.6%)

4.1.2. InceptionV3

Showing the same pattern as DenseNet201, the model gets overfitted on the training set in both cases (training the model from scratch or using transfer learning) but using weights from the model trained on ImageNet can make the model achieve better results on the validation set (79.9% vs 77.6%). The results are shown in Figure 4-2. It might be worth mentioning that the validation accuracy does not improve significantly by fine-tuning the model, showing that the model was already performing at a peak level just by training on a different space. The model trained from scratch shows signs of overfitting from 4th epoch. Eventually both cases achieve an accuracy of over 95% on the training set.

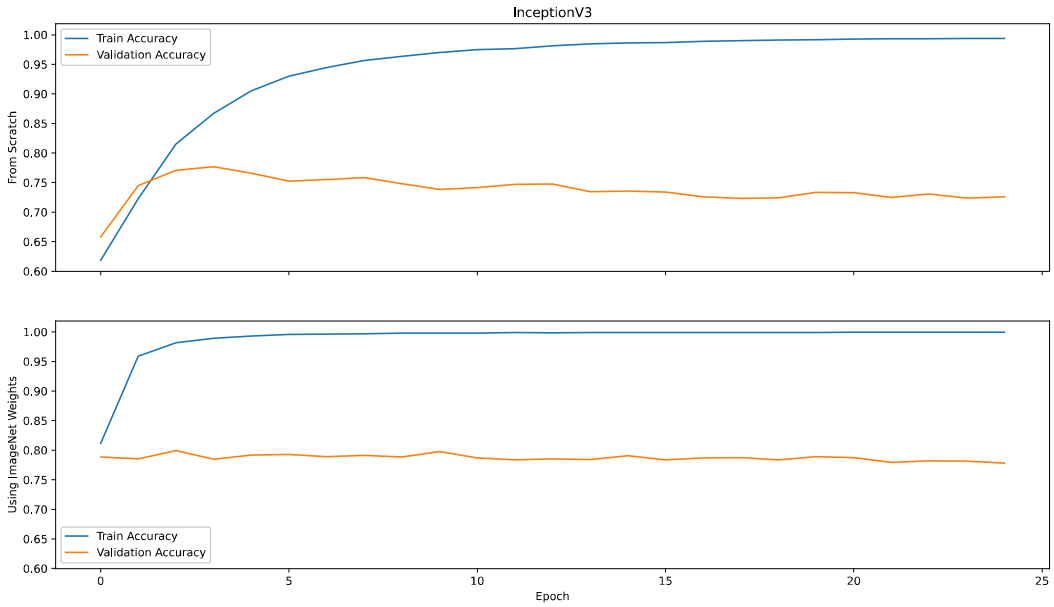

Figure 4-2: Training InceptionV3 model from scratch (Highest validation accuracy: 77.6) vs using pre-trained weights and fine-tuning the model (Highest validation accuracy: 79.9%)

4.1.3. Xception

Even though there is not a significant difference between learning from scratch and using the ImageNet weights, using the weights from a model trained on ImageNet is more resilient to overfitting and in the end (validation accuracy does not drop as the training goes on), can achieve higher accuracies as shown in Figure 4-3.

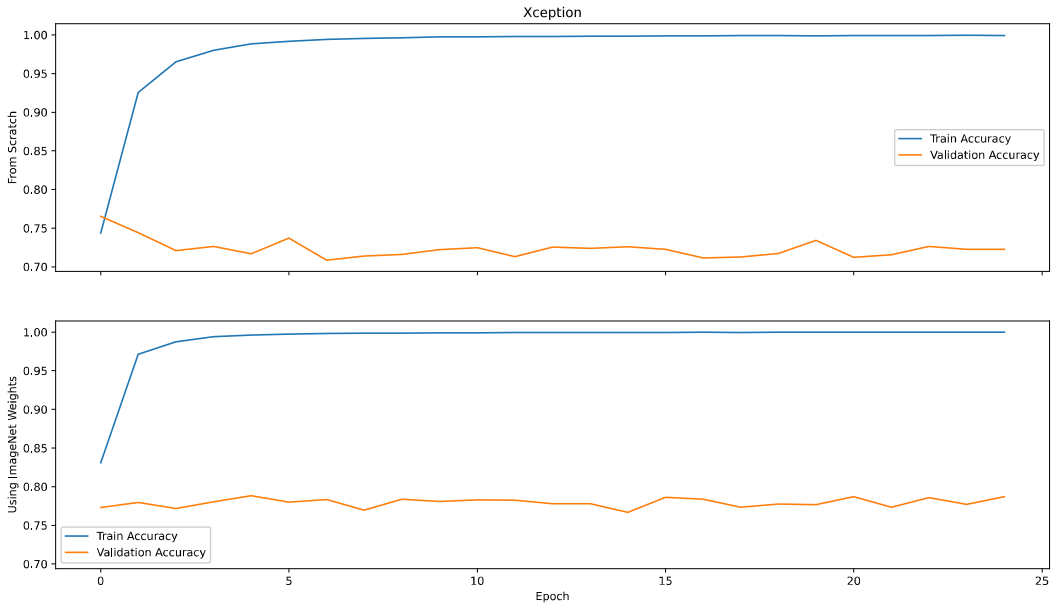

Figure 4-3: Training Xception model from scratch (Highest validation accuracy: 76.5%) vs using pre-trained weights and fine-tuning the model (Highest validation accuracy: 78.7%)

4.2. Transfer Learning from Sound Classification Models

Using the VGGish model trained on AudioSet and fine-tuning the results on ESC-50 Dataset vs training the same model on ESC-50 Dataset from scratch, the effect of transferring knowledge from the sound classification domain is explored. Figure 4-4 shows how using transfer learning makes the training process more resilient to over fitting and achieves a higher validation accuracy.

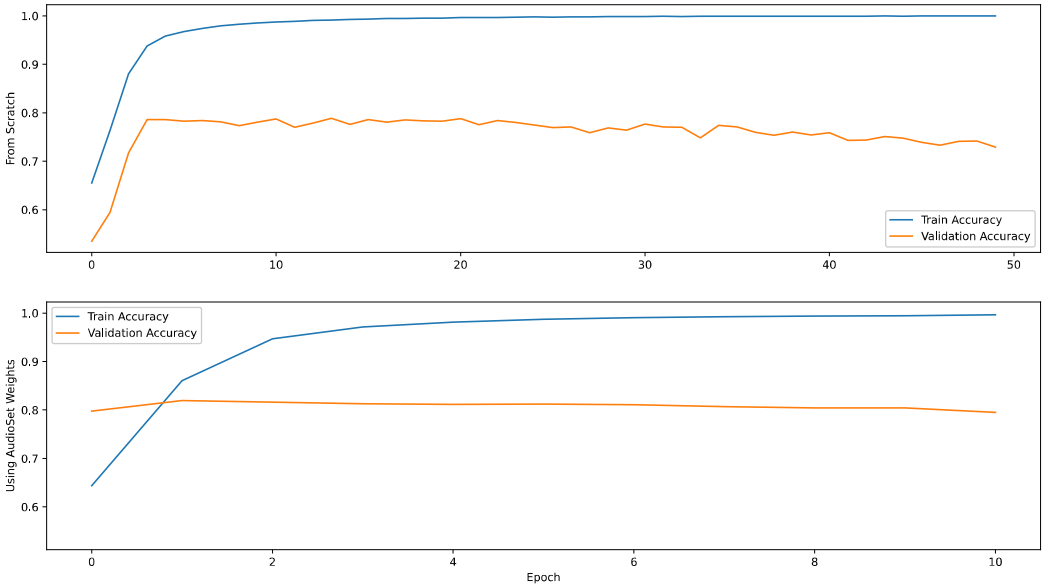

Figure 4-4: Training VGGish model from scratch (Highest validation accuracy: 78.8%) vs using pre-trained weights and fine-tuning the model (Highest validation accuracy: 81.9%)

Although the VGGish model trained from scratch achieves good scores, using the pre-trained model on AudioSet and fine-tuning it, the performance on the validation set gets an 81.9% accuracy which is the highest achieved at this point with this configuration.

4.3. Cosine Similarity vs a Neural Network as Comparison Function

Using a pre-trained VGGish model on AudioSet as the feature extractor, the effect of having a Neural Network or cosine similarity as the comparison function is explored. Figure 4-5 compares the results from using cosine similarity and the proposed Neural Network in Figure 3-13 on ESC-50 dataset.

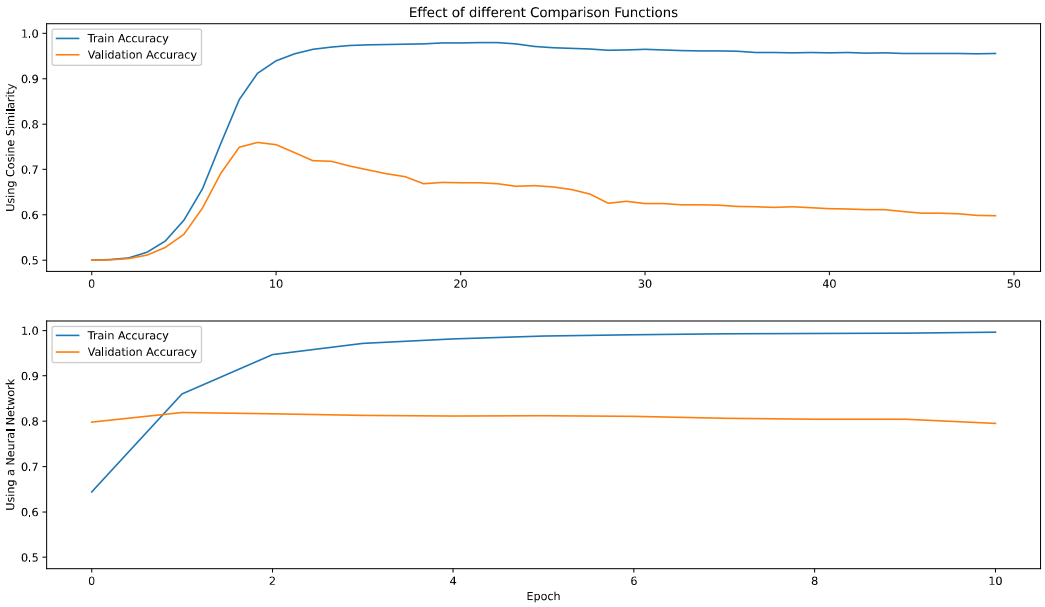

Figure 4-5: Training VGGish model with Cosine Similarity as Comparison Function (Highest validation accuracy: 75.9%) vs using a Neural Network as Comparison Function (Highest validation accuracy: 81.9%)

The results show a great improvement (6% better accuracy on the validation set) made by using a Neural Network as the comparison function compared to a cosine similarity.

4.4. Classification Task

From this point forward, all tests are done using a fine-tuned VGGish network trained on AudioSet as the feature extractor and a Neural Network as comparison function. Each audio in ESC50 Dataset is 5s long and is converted into 7 frames for analysis. The classification is done based on the mean score of all these 7 frames for each class.

4.4.1. Effect of choosing different sets of classes for testing

As the human ear might find it difficult to distinguish between sounds such as rain vs. sea wave, a One/Few Shot Learning Model would achieve different accuracies on different class sets. To find a general accuracy, using ESC-10 with 10 classes as a

test set and for a k-way classification, every possible combination of k classes out of 10 classes were tested and an average result is calculated.

Figure 4-6 shows the normalized classification accuracy result for a One-Shot 3-Way classification. Figure 4-7 shows the same results for a One-Shot 5-Way classification.

Figure 4-6 to Figure 4-11, Figure 4-13 and Figure 4-14 all show the confusion matrices based on average values. There are 10 classes in the test set which means for the 3-way classification tests, 120 different combinations and for the 5-way classification tests, 252 different combinations can be chosen from the available 10. Each of these combinations is tested separately resulting in a separate confusion matrix, but instead of displaying 120 confusion matrices for 3-way classification and 252 confusion matrices for the 5-way classification, the results of the 3-way and 5-way classifications are combined into a single figure. To create these average confusion matrices, the results of all the combinations for 3-way or 5-way classification are combined, and each cell is calculated by finding the percentage of all the tests that a true label (row) was predicted as a specific value (column) by the model.

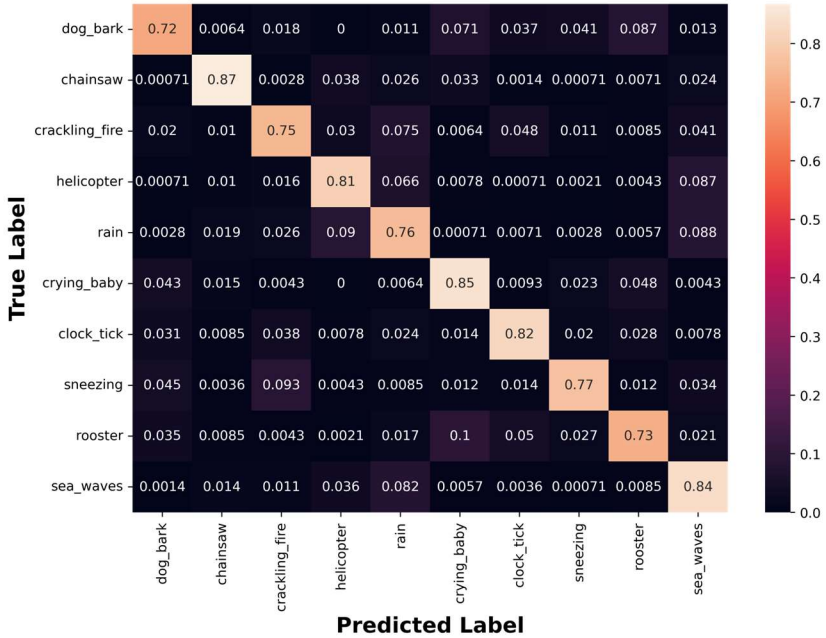

Figure 4-6: A One-Shot 3-Way classification normalized result. Each cell represents the percentage that the row class was classified as the column class (column) – Average Accuracy: 79.2%

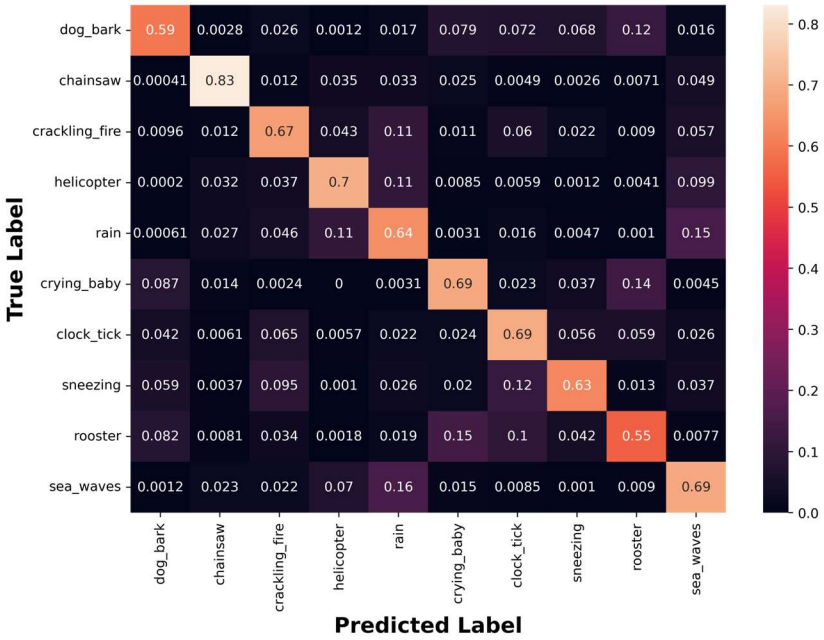

Figure 4-7: A One-Shot 5-Way classification normalized result. Each cell represents the percentage that the row class was classified as the column class (column) – Average Accuracy: 66.8%

These results show how a selective group of distinguishable classes can result in much higher accuracy, so in order to find a general answer the same testing must be performed on all possible combinations in the test set. For example, this model has a problem distinguishing **rain** sound from **sea_wave** and **helicopter**, but barely mistakes **rain** for **dog_bark** which also makes sense as many humans may have the same difficulty.

4.4.2. Few Shot Learning with average similarity

Instead of classifying the input based on similarity with one sample of each class, the classification is done based on average similarity to 5 samples. Figure 4-8 shows the results for doing a Few Shot classification with 5 samples and 3 classes (5-Shot 3-Way) and can be compared to Figure 4-6 where the same test is done with only one sample (One-Shot 3-Way). Figure 4-9 shows the result for 5-Shot 5-Way classification and can be compared to Figure 4-7 which is a One-Shot 5-Way classification.

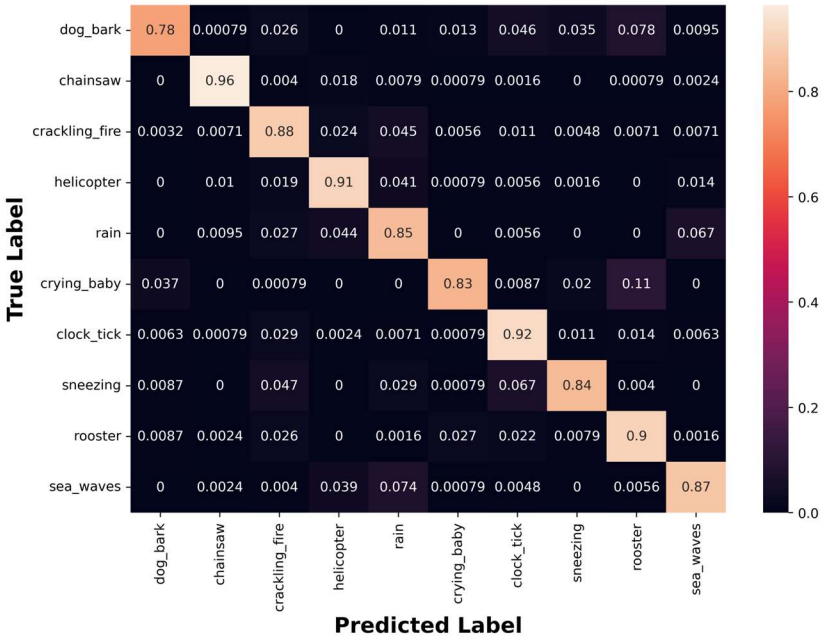

Figure 4-8: A 5-Shot 3-Way classification normalized result. Each cell represents the percentage that the row class was classified as the column class (column) – Average Accuracy: 87.4%

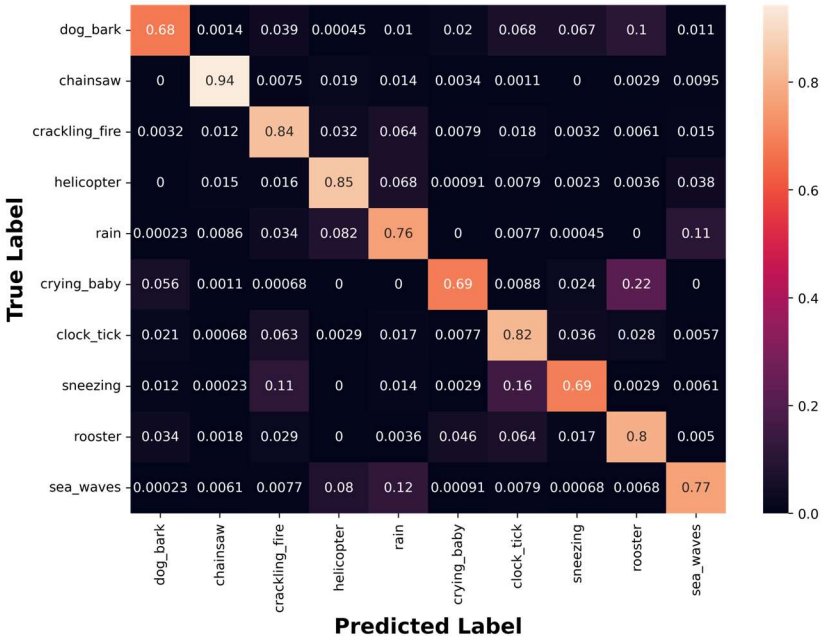

Figure 4-9: A 5-Shot 5-Way classification normalized result. Each cell represents the percentage that the row class was classified as the column class (column) – Average Accuracy: 78.4%

Using the same model and by having 5 samples instead of one, the average accuracy for the 3-way classification has improved by 8.2% and for the 5-way

classification, it has improved by 11.6% which shows how having more samples can significantly improve the accuracy.

4.4.3. Few Shot Learning with the Super Sample

Using the same feature extractor and the same Neural Network as comparison function, 5 labeled samples of each class are converted into one super sample (mean of five feature vectors representing five labeled sample) and the same tests are performed. Figure 4-10 shows the results for doing a Few Shot classification with 5 samples converted to a super sample and 3 classes (5-Shot 3-Way) and can be compared with the results shown in Figure 4-8 (same test without creating the super sample). Figure 4-11 also shows a Few Shot Classification with 5 samples converted to a super sample with 5 classes (5-Shot 5-Way) and can be compared with Figure 4-9 (same test without creating the super sample).

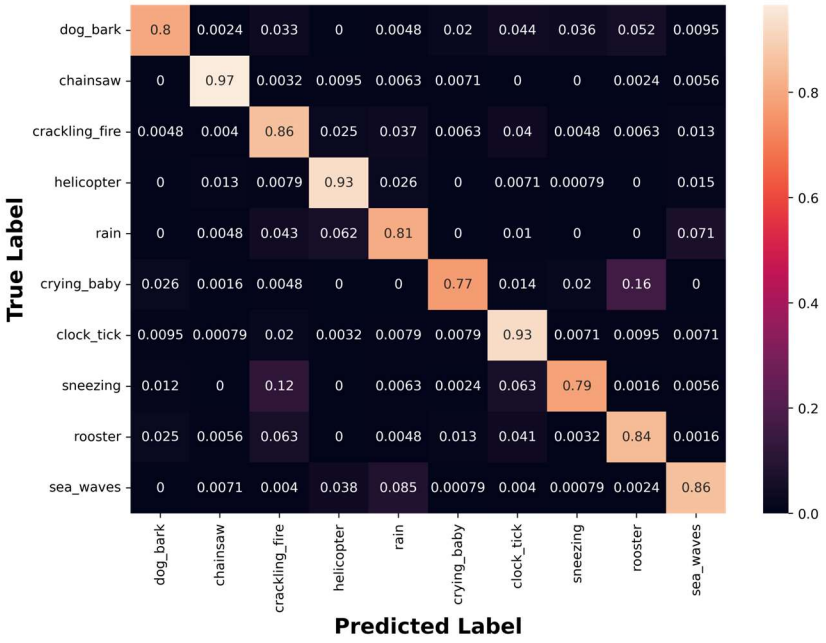

Figure 4-10: A 5-Shot 3-Way classification using super sample normalized result. Each cell represents the percentage that the row class was classified as the column class (column) – Average Accuracy: 85.6%

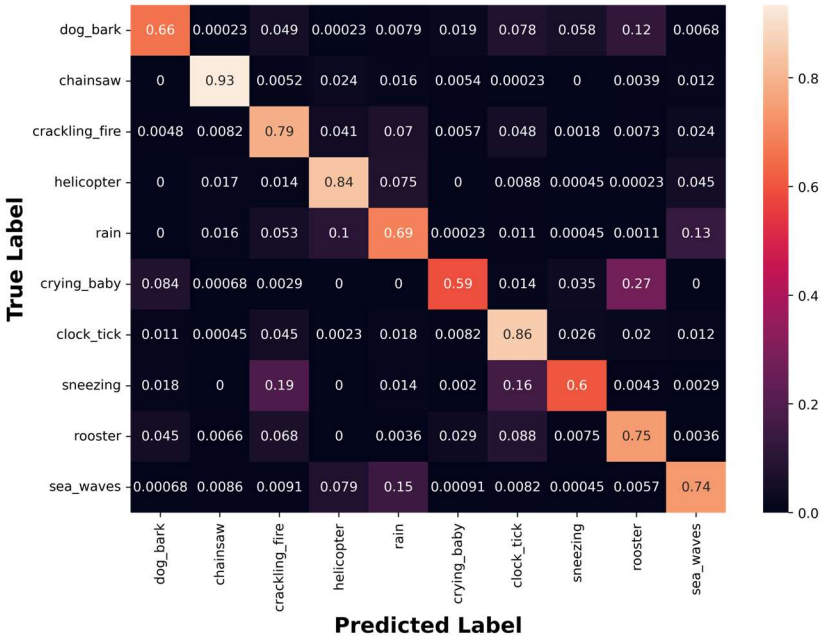

Figure 4-11: A 5-Shot 5-Way classification using super sample normalized result. Each cell represents the percentage that the row class was classified as the column class (column) – Average Accuracy: 74.5%

Using the super sample, the accuracy has decreased compared to looking at each labeled sample separately, however, it has improved the results compared to just having one sample for each class. This suggests that the super sample is a better representation of a class compared to a single random labeled sample. The results for these 3 different One/Few Shot Learning methods are summarized in Table 6.

Table 6: Summary of One-Shot, super sample and Few Shot Learning accuracies created using the same feature extractor and the same comparison network

Number of classes	One-Shot	Super Sample	Few Shot
3-Way	79.2%	85.6%	**87.4%**
5-Way	66.8%	74.5%	**78.4%**

Using the t-distributed stochastic neighbor embedding (t-SNE) method [79] (a way of displaying high dimensional data in a 2D or 3D space), a representation of different samples in the feature space is plotted to see how different samples of each class are located compared to each other.

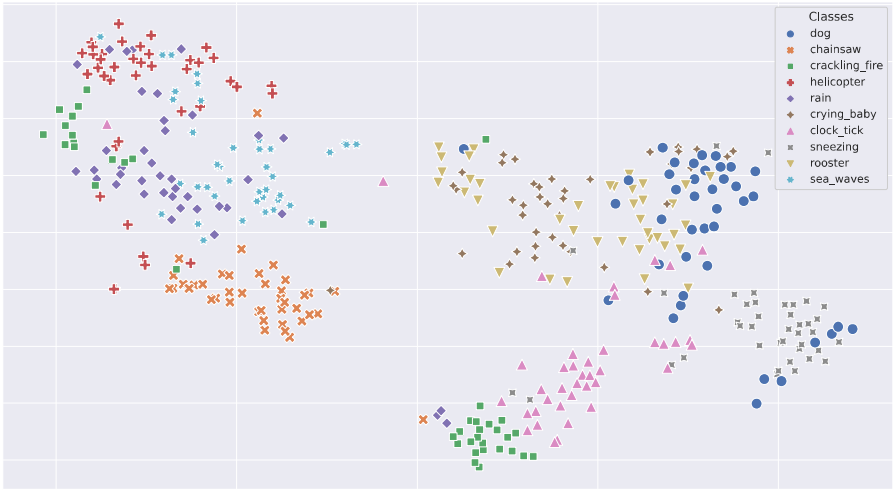

Figure 4-12: A 2D representation of ESC-10 Dataset in the feature space created by VGGish feature extractor

This plot shows how some classes like **chainsaw** and **rooster** are well separated in the feature space while some classes like **rain** and **sea_waves** are intertwined.

Plotting the feature space in 2D using t-SNE gives valuable insight on how the feature extractor is performing. In the current work, this method was used only during the latest stages of the development, however, it is recommended that any future work on One/Few Shot Learning leverages this approach from the first time a feature vector is calculated.

4.4.4. Converting multiple frames into a new feature vector

Using the novel 2-layer feature extraction for audio classification with the first layer being the VGGish feature extractor and the second layer being the LSTM Auto Encoder, the same classification task has been investigated. For these tests, instead of using the mean of similarity scores of different samples, the following function has been used,

$$Probability = \frac{1}{2} * (mean(s) + \max(s)) \quad (9)$$

where **s** is the set of the scores the input gets in comparison with each labeled sample separately.

This approach divides the similarity score between the average similarity of the input with different labeled samples and the maximum similarity value it got. Therefore, in comparison of similar classes, a class would be chosen when it has a labeled sample very similar to the input and also other samples from that class share some similarities with the input.

Figure 4-13 shows the results for doing a 5-Shot 3-Way Classification using the two-layer feature extraction method and can be compared with the results shown in Figure 4-8 (same test without the second feature extraction layer). Figure 4-14 also shows a 5-Shot 5-Way Classification using the two-layer feature extraction method and can be compared with Figure 4-9 (same test without the second feature extraction layer).

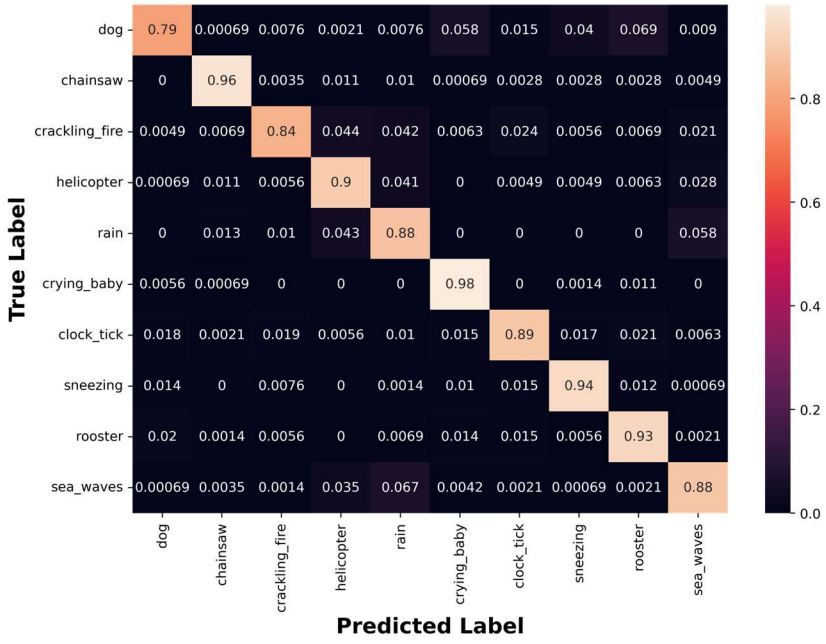

Figure 4-13: A 5-Shot 3-Way classification using two-layer feature extraction with LSTM Auto Encoder normalized result. Each cell represents the percentage that the row class was classified as the column class (column) – Average Accuracy: 89.8%

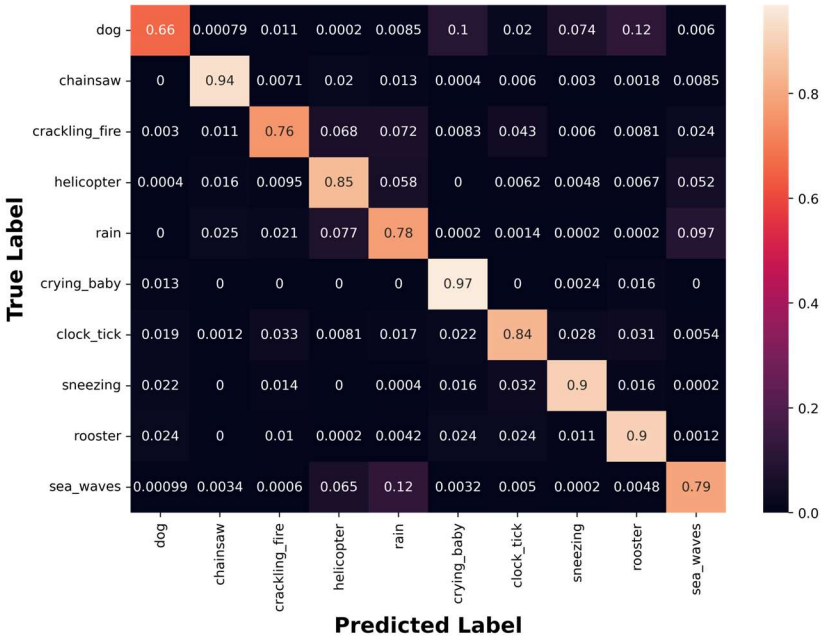

Figure 4-14: A 5-Shot 5-Way classification using two-layer feature extraction with LSTM Auto Encoder normalized result. Each cell represents the percentage that the row class was classified as the column class (column) – Average Accuracy: 83.9%

To summarize, Table 7 shows the results on how using a second layer of feature extraction and looking at how the sound has changed over multiple frames (7 frames in these tests) can help the model achieve higher accuracies.

Table 7: A comparison between classification accuracies of a normal Few Shot Learning vs. the proposed Two-Layer Feature Extraction Few Shot Learning

Number of classes	Few Shot	Two-Layer Feature Extraction (LSTM Auto Encoder)
3-Way	87.4%	**89.8%**
5-Way	78.4%	**83.9%**

Using the t-SNE, a new representation of different samples in the feature space is plotted in Figure 4-15 to see how different samples of each class are located compared to each other.

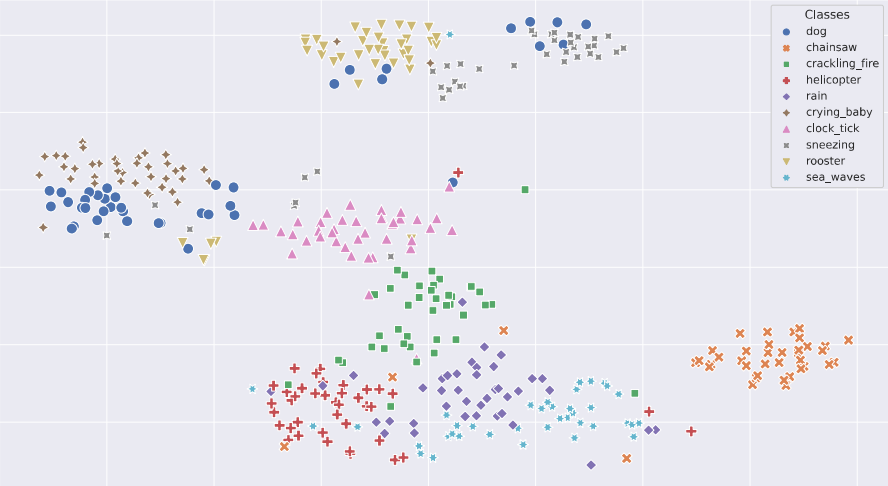

Figure 4-15: A 2D representation of ESC-10 Dataset in the feature space created by two layers of feature extractor: VGGish feature extractor and LSTM Auto Encoder

Comparison of Figure 4-15 with Figure 4-12 which was created by just using the VGGish feature extractor shows that most classes are much more distinguishable from each other in feature space using the two-layer feature extraction method. It also shows that the second layer of feature extractor has improved the model in a way that it would be able to classify sound with higher accuracies.

Chapter 5. Discussion and Conclusion

This research work aimed to develop a machine learning model that could identify the aversive sounds for people with Autism Spectrum Disorder (ASD) based on their specific needs and sound sensitivities. Artificial Intelligence systems, specifically the ones using the approach known as One/Few-shot learning were investigated in the current work and to see if knowledge transfer from image classification (transfer learning) improves the accuracy of a One/Few Shot Learning model (fine-tuned for sound classification), three different image classification models (pre-trained on ImageNet) were fine-tuned for One/Few Shot Learning. Through multiple training and testing scenarios, it is shown that transferring knowledge will help the model find a better general answer (helps with the overfitting problem), leading to better performance on the validation set.

In addition, the idea of transfer learning from an audio classification model to the feature extractor of a One/Few Shot Learning model was explored by fine-tuning the VGGish model (A derivative of the VGG model used for audio classification, pre-trained on AudioSet) for One/Few Shot Learning and benchmarked against a comparable one trained from scratch. The results showed that transfer learning from the pre-trained model on AudioSet can achieve better accuracy and be less prone to over-fitting. Compared to transfer learning from image classification, the VGGish model pre-trained on AudioSet achieved even higher accuracy.

As Siamese Network (the One/Few Shot Learning implementation used in this research) needs a comparison function for comparing the input with the labeled sample (in feature space), a neural network was proposed and trained to compare the outputs of the feature extractors. The test results of the proposed neural network (called the comparison network) showed an improvement in the model's accuracy compared to using cosine distance (a common approach for Siamese Networks) for comparing the input and the labeled sample.

Eventually, to create a model that looks at more than one time-frame, an extra layer of feature extraction was proposed using LSTM Auto Encoders and the tests showed a significant improvement in the accuracy of the new two-layer feature extraction One/Few Shot Learning compared to the same model without the extra

feature extraction layer. The proposed model achieved the highest score in all the tests done here and beat the highest accuracy reported on an environmental sound Few-Shot Learning Classification (5-Shot 5-Way Classification, 83.9% vs 78.3% [33]).

5.1. Future works

First, the representations used to create the three channels for image classification models are the most common audio representations and they work perfectly when used separately. But the combination of these three representations should be further investigated to see if it has the same correlation between the channels as an RGB image. Changing the representations in a way that the combination becomes more similar to how the three channels of an RGB image are correlated, could result in a much better performance when transferring knowledge from image classification. Also, the gaussian wavelet used for CWT is a bell-shaped symmetrical wavelet which makes it a great general mother wavelet, but other options such as Mexican Hat or Morlet wavelet should also be considered and tested.

Second, a newly published model, EfficientNet [80], has shown state-of-the-art performance on ImageNet classification. Using a similar approach used in designing the VGGish model from the original VGG model, EfficientNet can probably be optimized to work only with the Mel-Spectrogram of audio for audio classification. That model should be trained for audio classification on AudioSet and then, using transfer learning, the same steps done in this thesis should be repeated and hopefully, a higher performance can be achieved.

Third, the proposed LSTM Auto Encoder can still be investigated for improvement. The designed LSTM Auto Encoder made a great improvement to the One/Few Shot Learning model, but the architecture of the LSTM Auto Encoder was not fully investigated and probably has room for improvement.

References

[1] H. Hodges, C. Fealko, and N. Soares, “Autism spectrum disorder: definition, epidemiology, causes, and clinical evaluation,” *Transl. Pediatr.*, vol. 9, no. Suppl 1, p. S55, Feb. 2020, doi: 10.21037/TP.2019.09.09.

[2] J. Baio *et al.*, “Prevalence of Autism Spectrum Disorder Among Children Aged 8 Years - Autism and Developmental Disabilities Monitoring Network, 11 Sites, United States, 2014,” *MMWR. Surveill. Summ.*, vol. 67, no. 6, pp. 1–23, 2018, doi: 10.15585/MMWR.SS6706A1.

[3] S. D. Tomchek and W. Dunn, “Sensory processing in children with and without autism: a comparative study using the short sensory profile,” *Am. J. Occup. Ther.*, vol. 61, no. 2, pp. 190–200, 2007, doi: 10.5014/AJOT.61.2.190.

[4] E. Gomes, N. T. Rotta, F. S. Pedroso, P. Sleifer, and M. C. Danesi, “Auditory hypersensitivity in children and teenagers with autistic spectrum disorder,” *Arq. Neuropsiquiatr.*, vol. 62, no. 3 B, pp. 797–801, 2004, doi: 10.1590/S0004-282X2004000500011.

[5] K. O’Connor, “Auditory processing in autism spectrum disorder: a review,” *Neurosci. Biobehav. Rev.*, vol. 36, no. 2, pp. 836–854, Feb. 2012, doi: 10.1016/J.NEUBIOREV.2011.11.008.

[6] Z. J. Williams, E. Suzman, and T. G. Woynaroski, “Prevalence of Decreased Sound Tolerance (Hyperacusis) in Individuals with Autism Spectrum Disorder: A Meta-Analysis,” *Ear Hear.*, pp. 1137–1150, 2021, doi: 10.1097/AUD.0000000000001005.

[7] E. K. Jones, M. Hanley, and D. M. Riby, “Distraction, distress and diversity: Exploring the impact of sensory processing differences on learning and school life for pupils with autism spectrum disorders,” *Res. Autism Spectr. Disord.*, vol. 72, p. 101515, Apr. 2020, doi: 10.1016/J.RASD.2020.101515.

[8] N. E. Scheerer, T. Q. Boucher, B. Bahmei, G. Iarocci, S. Arzanpour, and E. Birmingham, “Family Experiences of Decreased Sound Tolerance in ASD,” *J.*

Autism Dev. Disord., 2021, doi: 10.1007/S10803-021-05282-4.

[9] K. Fackrell *et al.*, “Identifying and prioritising unanswered research questions for people with hyperacusis: James Lind Alliance Hyperacusis Priority Setting Partnership,” *BMJ Open*, vol. 9, no. 11, p. e032178, Nov. 2019, doi: 10.1136/BMJOPEN-2019-032178.

[10] I. Jager, P. de Koning, T. Bost, D. Denys, and N. Vulink, “Misophonia: Phenomenology, comorbidity and demographics in a large sample,” *PLoS One*, vol. 15, no. 4, p. e0231390, Apr. 2020, doi: 10.1371/JOURNAL.PONE.0231390.

[11] A. Woodhouse and P. d. Drummond, “Mechanisms of Increased Sensitivity to Noise and Light in Migraine Headache,” *Cephalalgia*, vol. 13, no. 6, pp. 417–421, Nov. 1993, doi: 10.1046/j.1468-2982.1993.1306417.x.

[12] D. M. Baguley and D. J. McFerran, “Hyperacusis and disorders of loudness perception,” *Textb. Tinnitus*, pp. 13–23, 2011, doi: 10.1007/978-1-60761-145-5_3/COVER/.

[13] L. N. Stiegler and R. Davis, “Understanding Sound Sensitivity in Individuals with Autism Spectrum Disorders,” *Focus Autism Other Dev. Disabl.*, vol. 25, no. 2, pp. 67–75, 2010, doi: 10.1177/1088357610364530.

[14] O. K. Toffa and M. Mignotte, “Environmental sound classification using local binary pattern and audio features collaboration,” *IEEE Trans. Multimed.*, vol. 23, pp. 3978–3985, 2021, doi: 10.1109/TMM.2020.3035275.

[15] B. Bahmei, E. Birmingham, and S. Arzanpour, “CNN-RNN and Data Augmentation Using Deep Convolutional Generative Adversarial Network for Environmental Sound Classification,” *IEEE Signal Process. Lett.*, vol. 29, pp. 682–686, 2022, doi: 10.1109/LSP.2022.3150258.

[16] Li Fe-Fei, Fergus, and Perona, “A Bayesian approach to unsupervised one-shot learning of object categories,” in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp. 1134–1141 vol.2, doi: 10.1109/ICCV.2003.1238476.

[17] G. Koch, R. Zemel, R. S.-I. deep learning workshop, and undefined 2015, "Siamese neural networks for one-shot image recognition," *cs.toronto.edu*, Accessed: Jun. 17, 2022. [Online]. Available: http://www.cs.toronto.edu/~gkoch/files/msc-thesis.pdf.

[18] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June-2015, pp. 815–823, Mar. 2015, doi: 10.1109/cvpr.2015.7298682.

[19] S. Ravindran and D. V. Anderson, "Audio classification and scene recognition and for hearing aids," *Proc. - IEEE Int. Symp. Circuits Syst.*, pp. 860–863, 2005, doi: 10.1109/ISCAS.2005.1464724.

[20] X. Zhang, Y. Zou, and W. Shi, "Dilated convolution neural network with LeakyReLU for environmental sound classification," *Int. Conf. Digit. Signal Process. DSP*, vol. 2017-August, Nov. 2017, doi: 10.1109/ICDSP.2017.8096153.

[21] N. Sengupta, M. Sahidullah, and G. Saha, "Lung sound classification using cepstral-based statistical features," *Comput. Biol. Med.*, vol. 75, pp. 118–129, Aug. 2016, doi: 10.1016/J.COMPBIOMED.2016.05.013.

[22] Li Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, 2006, doi: 10.1109/TPAMI.2006.79.

[23] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots, "One-Shot Learning for Semantic Segmentation," *Br. Mach. Vis. Conf. 2017, BMVC 2017*, Sep. 2017, doi: 10.48550/arxiv.1709.03410.

[24] M. Woodward, C. Finn, and B. A. Research, "Active One-shot Learning," Feb. 2017, doi: 10.48550/arxiv.1702.06559.

[25] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching Networks for One Shot Learning." 2017.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-

scale hierarchical image database," pp. 248–255, Mar. 2010, doi: 10.1109/CVPR.2009.5206848.

[27] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science (80-.)*., vol. 350, no. 6266, pp. 1332–1338, Dec. 2015, doi: 10.1126/SCIENCE.AAB3050/SUPPL_FILE/LAKE-SM.PDF.

[28] S. Chanda, A. C. Gv, A. Brun, A. Hast, U. Pal, and D. Doermann, "Face recognition - A one-shot learning perspective," *Proc. - 15th Int. Conf. Signal Image Technol. Internet Based Syst. SISITS 2019*, pp. 113–119, Nov. 2019, doi: 10.1109/SITIS.2019.00029.

[29] I. Vélez, C. Rascon, and G. Fuentes-Pineda, "One-Shot Speaker Identification for a Service Robot using a CNN-based Generic Verifier." 2018.

[30] P. Wolters, C. Careaga, B. Hutchinson, and L. Phillips, "A Study of Few-Shot Audio Classification," Dec. 2020, doi: 10.48550/arxiv.2012.01573.

[31] J. Salamon, C. Jacoby, and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research," in *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014, pp. 1041–1044, doi: 10.1145/2647868.2655045.

[32] J. Pons, J. Serra, and X. Serra, "Training neural audio classifiers with few data," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2019-May, pp. 16–20, Oct. 2018, doi: 10.48550/arxiv.1810.10274.

[33] S. Zhang, Y. Qin, K. Sun, and Y. Lin, "Few-shot audio classification with attentional graph neural networks," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2019-September, pp. 3649–3653, 2019, doi: 10.21437/INTERSPEECH.2019-1532.

[34] Y. L. E. S. J. Bromley I. Guyon and R. Shah, "Signatureverification using a 'siamese' time delay neural network," *Proc. Adv.Neural Inf. Process. Syst , pp. 737–744.*, 1994.

[35] K. Palanisamy, D. Singhania, and A. Yao, "Rethinking CNN Models for Audio

Classification," Jul. 2020, Accessed: May 20, 2022. [Online]. Available: https://arxiv.org/abs/2007.11154v2.

[36] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, doi: 10.1109/TPAMI.2016.2644615.

[37] A. Majkowska *et al.*, "Chest radiograph interpretation with deep learning models: Assessment with radiologist-adjudicated reference standards and population-adjusted evaluation," *Radiology*, vol. 294, no. 2, pp. 421–431, Dec. 2020, doi: 10.1148/RADIOL.2019191293/ASSET/IMAGES/LARGE/RADIOL.2019191293.FIG5B.JPEG.

[38] J. F. Gemmeke *et al.*, "Audio Set: An ontology and human-labeled dataset for audio events," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 776–780, Jun. 2017, doi: 10.1109/ICASSP.2017.7952261.

[39] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," 2011.

[40] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," *Proc. 18th Int. Soc. Music Inf. Retr. Conf. ISMIR 2017*, pp. 141–149, Mar. 2017, doi: 10.48550/arxiv.1703.09179.

[41] H. Xie and T. Virtanen, "Zero-Shot Audio Classification Based On Class Label Embeddings," *IEEE Work. Appl. Signal Process. to Audio Acoust.*, vol. 2019-October, pp. 264–267, Oct. 2019, doi: 10.1109/WASPAA.2019.8937283.

[42] G. Gwardys and D. Grzywczak, "Deep Image Features in Music Information Retrieval," *INTL J. Electron. Telecommun.*, vol. 60, no. 4, pp. 321–326, 2014, doi: 10.2478/eletel-2014-0042.

[43] J. Salamon, J. B.-I. S. P. Letters, and undefined 2017, "Deep convolutional neural networks and data augmentation for environmental sound classification," *ieeexplore.ieee.org*.

[44] S. Adapa, “Urban Sound Tagging using Convolutional Neural Networks,” pp. 5–9, Sep. 2019, doi: 10.48550/arxiv.1909.12699.

[45] E. Kazakos, A. Nagrani, A. Zisserman, and Di. Damen, “EPIC-Fusion: Audio-Visual Temporal Binding for Egocentric Action Recognition,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-October, pp. 5491–5500, Aug. 2019, doi: 10.48550/arxiv.1908.08498.

[46] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nat. 1986 3236088*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.

[47] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3104–3112, Sep. 2014, doi: 10.48550/arxiv.1409.3215.

[48] N. Srivastava, E. Mansimov, and R. Salakhutdinov, “Unsupervised Learning of Video Representations using LSTMs,” *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 1, pp. 843–852, Feb. 2015, doi: 10.48550/arxiv.1502.04681.

[49] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection.”

[50] M. Meyer, J. Beutel, and L. Thiele, “Unsupervised Feature Learning for Audio Analysis,” *5th Int. Conf. Learn. Represent. ICLR 2017 - Work. Track Proc.*, Dec. 2017, doi: 10.48550/arxiv.1712.03835.

[51] J. Han and C. Moraga, “The influence of the sigmoid function parameters on the speed of backpropagation learning,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 930, pp. 195–201, 1995, doi: 10.1007/3-540-59497-3_175.

[52] Y. LeCun *et al.*, “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/NECO.1989.1.4.541.

[53] “Recent Advances in Deep Learning for Speech Research at Microsoft - Microsoft

Research.” https://www.microsoft.com/en-us/research/publication/recent-advances-in-deep-learning-for-speech-research-at-microsoft/ (accessed May 12, 2022).

[54] S. Hershey *et al.*, “CNN Architectures for Large-Scale Audio Classification,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 131–135, Sep. 2016, doi: 10.1109/ICASSP.2017.7952132.

[55] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” *IEEE Trans. Audio, Speech Lang. Process.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014, doi: 10.1109/TASLP.2014.2339736.

[56] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 1, pp. 448–456, Feb. 2015, doi: 10.48550/arxiv.1502.03167.

[57] J. Kukačka, V. Golkov, and D. Cremers, “Regularization for Deep Learning: A Taxonomy,” Oct. 2017, doi: 10.48550/arxiv.1710.10686.

[58] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.

[59] H. Fan, M. Jiang, L. Xu, H. Zhu, J. Cheng, and J. Jiang, “Comparison of Long Short Term Memory Networks and the Hydrological Model in Runoff Simulation,” *Water 2020, Vol. 12, Page 175*, vol. 12, no. 1, p. 175, Jan. 2020, doi: 10.3390/W12010175.

[60] K. J. Piczak, “ESC: Dataset for Environmental Sound Classification,” in *Proceedings of the 23rd ACM International Conference on Multimedia*, 2015, pp. 1015–1018, doi: 10.1145/2733373.2806390.

[61] M. Huzaifah, “Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks.” 2017.

[62] C. H. (Chi-hau) Chen, “Pattern recognition and artificial intelligence : proceedings of the Joint Workshop on Pattern Recognition and Artificial Intelligence, held at

Hyannis, Massachusetts, June 1-3, 1976," p. 621, Accessed: Nov. 30, 2022. [Online]. Available: https://books.google.com/books/about/Pattern_Recognition_and_Artificial_Intel.html?id=wW9QAAAAMAAJ.

[63] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," Accessed: Aug. 05, 2022. [Online]. Available: https://www.ams.org/journal-terms-of-use.

[64] A. V. Oppenheim and R. W. Schafer, "From frequency to quefrency: A history of the cepstrum," *IEEE Signal Process. Mag.*, vol. 21, no. 5, 2004, doi: 10.1109/MSP.2004.1328092.

[65] Y. Gong, Y. A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 1, pp. 56–60, Apr. 2021, doi: 10.48550/arxiv.2104.01778.

[66] A. Kumar and V. K. Ithap, "A Sequential Self Teaching Approach for Improving Generalization in Sound Event Recognition," *37th Int. Conf. Mach. Learn. ICML 2020*, vol. PartF168147-7, pp. 5403–5413, Jun. 2020, doi: 10.48550/arxiv.2007.00144.

[67] P. Lopez-Meyer, J. A. Del Hoyo Ontiveros, H. Lu, and G. Stemmer, "Efficient end-to-end audio embeddings generation for audio classification on target applications," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2021-June, pp. 601–605, 2021, doi: 10.1109/ICASSP39728.2021.9414229.

[68] M. Mohaimenuzzaman, C. Bergmeir, I. T. West, and B. Meyer, "Environmental Sound Classification on the Edge: A Pipeline for Deep Acoustic Networks on Extremely Resource-Constrained Devices," Mar. 2021, doi: 10.48550/arxiv.2103.03483.

[69] B. Sievers, C. Parkinson, P. J. Kohler, J. M. Hughes, S. V. Fogelson, and T. Wheatley, "Visual and auditory brain areas share a representational structure that supports emotion perception," *Curr. Biol.*, vol. 31, no. 23, pp. 5192-5203.e4, Dec. 2021, doi: 10.1016/J.CUB.2021.09.043.

[70] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning Transferable Architectures for Scalable Image Recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 8697–8710, Jul. 2017, doi: 10.1109/CVPR.2018.00907.

[71] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 2261–2269, Aug. 2016, doi: 10.1109/CVPR.2017.243.

[72] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 2818–2826, Dec. 2015, doi: 10.1109/CVPR.2016.308.

[73] C. Szegedy *et al.*, “Going Deeper with Convolutions,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June-2015, pp. 1–9, Sep. 2014, doi: 10.1109/CVPR.2015.7298594.

[74] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 1800–1807, Oct. 2016, doi: 10.1109/CVPR.2017.195.

[75] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 770–778, Dec. 2015, doi: 10.1109/CVPR.2016.90.

[76] S. Abu-El-Haija *et al.*, “YouTube-8M: A Large-Scale Video Classification Benchmark,” Sep. 2016, Accessed: Jun. 02, 2022. [Online]. Available: http://arxiv.org/abs/1609.08675.

[77] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, Sep. 2014, doi: 10.48550/arxiv.1409.1556.

[78] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf.*

Track Proc., Sep. 2014, Accessed: Feb. 09, 2022. [Online]. Available: https://arxiv.org/abs/1409.1556v6.

[79] L. van der Maaten and G. Hinton, “Visualizing Data using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008, Accessed: Feb. 10, 2022. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html.

[80] M. Tan and Q. V. Le, “EfficientNetV2: Smaller Models and Faster Training,” Apr. 2021, doi: 10.48550/arxiv.2104.00298.