DATA-EFFICIENT APPROACHES FOR AUDIO CLASSIFICATION AND SEPARATION

BY

ZHEPEI WANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Doctoral Committee:

Professor Paris Smaragdis, Chair
Professor Svetlana Lazebnik
Professor Mark Hasegawa-Johnson
Professor Minje Kim, Indiana University

ABSTRACT

Recent advances in deep learning for computational audio processing are established upon sufficient annotated audio data. However, obtaining a substantial volume of high-quality annotations from in-the-wild audio remains a significant challenge. In this thesis, we propose and analyze data-efficient approaches for modeling audio signals to perform sound classification and separation. First, we present neural network architectures based on multi-dimensional unrolling of recurrent neural networks that allow the model to perform sound event detection with efficient usage of training data. Equipped with adaptive computation, the model further learns to intelligently adjust the amount of computation and enables processing when only partial information is available. Next, we propose approaches for recognizing sound classes under a time-varying distribution. We investigate continual learning techniques to train a classifier that can efficiently learn new sound classes without forgetting the past using generative replay. We further extend our approach to an unsupervised learning setup, where the model progressively learns representations for an indefinite number of sound classes with few labels presented. Last but not least, we investigate learning with limited annotated data using semi-supervised learning. We demonstrate the effectiveness of the proposed teacher-student framework on tasks including cross-modal audio-text representation learning, singing voice separation, and personalized speech enhancement. To this end, our proposed data-efficient algorithms for audio classification and source separation show high potential for reducing the labor cost for collecting high-quality annotated data, improving computational and storage efficiency, and enabling processing on memory-limited edge devices.

ACKNOWLEDGMENTS

Born and raised in southern China with mild winters and having spent four years of college in California with year-long sunshine, I could never imagine living under minus thirty degrees in the center of Illinois. Looking retrospectively, I could remember the excitement of seeing snowfalls for the first time in my life in the Siebel office and the days walking in inches of snow. I feel fortunate to survive through these harsh snowstorms, tornados, and the chaotic pandemic during my five-year journey as a Ph.D. student in Urbana-Champaign. I want to take this moment to thank my professors, my mentors, my peers and friends, and my family members for their support and for shaping the person I have become.

First and foremost, I would like to express my heartfelt gratitude to my advisor, Prof. Paris Smaragdis, for having me as a student and guiding me in my research, career, and life. I constantly felt clueless or frustrated about my research progress during the first few years, but nothing was more reassuring than having a conversation with Paris. I'm sincerely grateful for his patience in letting me take time to learn from trials and errors (even at the cost of me crashing the lab servers twice a day), explore various research topics, and develop my own research interest. One of my most precious moments in Urbana-Champaign is the tea excursions from Siebel to Green Street, filled with discussions ranging from ML-DSP to music, sports, and food. I've benefited from these conversations on how to become open-minded, communicate ideas, and interact with the audio research community.

Also, it is my great honor to have Prof. Svetlana Lazebnik, Prof. Mark Hasegawa-Johnson, and Prof. Minje Kim on my thesis committee. I would like to thank my committee for serving as inspirational role models in research and advising, and for providing constructive feedback on my thesis research.

During the course of my Ph.D., I've been honored to work with brilliant mentors. I would especially like to thank Cem (now Prof. Cem Subakan) for sharing his insights, trusting in me for my whimsical thoughts, and steering me back whenever I'm going south. I would never forget the days we craft projects together under extreme time pressure. I would also like to thank Dr. Ritwik Giri for our collaboration during my three internships at Amazon Web Services. Ritwik's dedication to audio research is inspirational, and his mentorship is indispensable for my growth as an independent researcher. It has also been my great pleasure to interact with and learn from exceptional scientists and pioneers in research for machine learning and digital signal processing, including Dr. Jean-Marc Valin, Dr. Umut Isik, Dr. Karim Helwani, Dr. Masahito Togami, Dr. Michael Goodwin, and Dr. Arvindh Krishnaswamy during the internship at AWS; Dr. Xiaolong Zhu and Dr. Shiyin Kang during the internship at Tencent; Prof. Mirco Ravanelli during the collaboration with MILA-Quebec; Prof. Tiago Tavares and Prof. Fabio Ayres during the collaboration with Insper.

In the meantime, I'm fortunate to develop strong friendships with a group of talented peers

along this journey. I want to thank my labmates: Shrikant, Thymios, Jonah, Ryley, Anny, Krishna, Dimitris, Dean, Junkai, and Xilin, for your support and collaboration. I would miss the whiteboard discussions after the group meetings, the around-the-world cuisines we have explored on campus, and the barbecue and volleyball in Paris' backyard. I'm also thankful to my friends Kedan, Jamshed, Hao, Kaiqi, Yansong, Xin, Yiran, Minjian, and Jiajun for sharing the enjoyable moments with me in Urbana-Champaign. It has also been a great pleasure to get to know young and talented researchers including Turab, Wo Jae, Manos, and Devansh during my internships at AWS.

My passion for computational audio research originates from my college days at Harvey Mudd. I'm sincerely grateful to Prof. TJ Tsai for offering me the opportunity for my first-ever hands-on experience in audio and deep learning research and for helping me develop disciplined and systematic research habits. I'd like to thank Prof. Ran Libeskind-Hadas, Prof. Zachary Dodds, and Prof. Yekaterina Kharitonova for their guidance. I'd also like to express my gratefulness towards my piano instructor, Prof. Hao Huang from Scripps College, not only for the tips on piano performance but also for the invaluable advice on learning to focus, developing the mentality for combatting hardships, and recognizing my own values.

During this unforgettable journey, I feel incredibly lucky to be accompanied by a group of long-lasting friends: Jacky, Kevin, James, Jackson, and Wending from Shenzhen Foreign Languages School; our 8F-gang (Andrew, Lincoln, and Tom), Kevin, Annie, and Xiwen from Shenzhen Middle School; Tony, Cam, Michael, Spencer, Jamie, and Cleo from the Claremont Colleges.

I'm sincerely grateful to Tony and Jackson for their genuine support during my toughest moments and for patiently listening to my concerns over the years. I'm deeply thankful to Anna for the companionship and for allowing me to open up about my vulnerabilities.

Last but not least, I'd reserve my utmost gratitude towards my mom, dad, and my grandparents for their unconditional support for me pursuing a Ph.D., for being my best role models of integrity and diligence, and for their boundless love throughout my 27 years of life. I owe my achievements today entirely to you.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

With the recent breakthroughs in computational hardware and the rising amount of data collection, deep learning has made a revolutionary impact on machine perception. Despite the extensive studies conducted on deep neural networks (DNNs) for computer vision [1, 2, 3] and natural language processing tasks [4, 5, 6, 7] over the past decade, the application of deep learning to audio processing tasks still remains a complex problem. One of the primary obstacles for audio processing tasks is the high dimensionality: for instance, a 10-second cellphone recording of a live concert may require an 80000-dimensional vector representation under a sampling rate of 8 kHz. Since audio signals are time series, another challenge for modeling audio is to learn the temporal dependency. Cues of auditory events may extend for several seconds or even minutes, which raises concerns for the model's ability and efficiency to capture long-term dependencies for analyzing sound scenes. Additionally, the learning process can be complicated with the overlapping of different sound events and the interference with ambient noise. This increases the difficulty to obtain high-quality annotations and necessitates further processing to separate individual sound events.

To overcome these difficulties, DNNs have significantly boosted the performance in a variety of audio processing tasks, including sound event detection [8, 9, 10, 11, 12], acoustic scene classification [13, 14], environmental sound classification [15, 16, 17], music genre classification [18, 19], speech emotion recognition [20, 21], keyword spotting [22], speaker verification [23], speech enhancement [24, 25] and separation [26, 27]. Despite the noticeable improvement, these computational models require a massive amount of annotated data to train, but obtaining large-scale labeled audio data with high quality and volume requires strenuous effort. To overcome the obstacles to data collection, training with efficient data usage has become an increasingly relevant problem for deep-learning-based audio applications. The primary goal of this thesis is to propose data-efficient algorithms for learning sound classifiers and separators.

1.1 STRUCTURE OF THE THESIS

In Chapter 1, we discuss the major challenges in data-driven audio processing tasks, including coverage of training data, completeness of information, data annotations, and time-varying distribution of data. Next, in Chapter 3, we propose two designs for neural network architectures that can deal with data constraints for sound event detection tasks. Then, in Chapter 4, we discuss continual learning algorithms for sound classification, where the model incrementally incorporates knowledge of sound events from new classes. Next, we present semi-supervised learning algorithms that effectively make use of unannotated data for sound classification and source separation tasks in Chapter 5. Finally, we conclude this thesis and discuss future directions and social impact in Chapter 6.

1.2 CHALLENGES FOR DATA-DRIVEN AUDIO PROCESSING TASKS

1.2.1 Limited Coverage of Training Data

The first problem we would want to resolve is the limited coverage of the training set. Even if large quantities of data samples are available, it is hard to obtain a comprehensive training set that can enclose possible real-world test cases, and hence the model that fails to generalize may overfit the distribution of the training data. Such concerns may arise in a multichannel audio classification task in a setting of the Internet of Things (IoT), where each channel corresponds to a single device with a microphone and the number of channels or devices at test time is unknown prior to inference. It is infeasible to collect a set of training recordings that covers all numbers of channels during test time; as a result, the model's test-time performance may degrade due to this shift in the number of channels between training and test time. To mitigate this issue, we introduce the Multi-View Network (MVN) [28, 29] (see Section 3.1) that can generalize to an unseen number of channels. By unrolling across the channel dimension, we leverage the property of recurrent neural networks (RNNs) to handle input sequences with an arbitrary number of channels, thereby making it possible to predict input with a different number of channels than training data.

1.2.2 Missing Information

The second challenge that audio classifiers may encounter is making predictions with partially missing information. Most classifiers are trained using spectrograms with a fixed number of bands as input; during inference, however, part of the frequency sub-bands might be unavailable to the model. For instance, test audio may be recorded from a device with limited bandwidth, hence bands with high-frequency components cannot be used for predictions. Additionally, data corruption may occur during the transmission or storage of the data, and the number of bands accessible to the model may vary across time frames. Nevertheless, for a variety of sound classes, the energy does not distribute evenly across frequencies, and hence it might be redundant to process all available sub-bands to predict the event classes. It is therefore desirable to train a classifier that intelligently determines the number of frequency sub-bands to consume and maintains a robust performance under a varying number of input sub-bands. We propose an adaptive frequency selection algorithm named HIDACT [30] (see Section 3.2), which extends the MVN with components for adjustable computation. Models learned with HIDACT are able to alter the number of sub-bands to process based on characteristics of input sound types without processing all frequency bands. It not only improves the robustness of the system under a mismatched number of frequency sub-bands between training and inference but also enhances computational efficiency.

1.2.3 Unavailable Data From the Past

Another common challenge that hinders model training in practice is that not all data samples can be collected at once. New data samples may arrive continually over time after the model is trained on an initial dataset. It is time and resource-consuming to retrain the model with new and old data combined. The problem becomes even more complicated when past data is not available for training due to damage to the hardware, limited storage capacity, or privacy regulations [31]. If updating the model's weights with new data naively, however, the model's performance is likely to deteriorate on the previous data, a phenomenon known as *catastrophic forgetting* [32]. To learn the model on a sequence of datasets or tasks with continuous shifts in distribution, a plethora of continual learning approaches have been studied [33, 34, 35, 36, 37, 38].

While these methods have been widely applied to image classification, we introduce continual learning to the audio domain in Chapter 4, which is a challenging task to model time-series input data in contrast to static data as images. In particular, in Section 4.1, we propose in [39] a generative replay framework to learn new sound classes while retaining knowledge of classes the model has previously seen. We train and update a generative network in parallel to the classifier with the same data stream, and use the generator to create a replay buffer without storing actual data from the past.

1.2.4 Data Without Annotations

Last but not least, we also study approaches to deal with audio data without annotations. For audio classification tasks, annotations refer to the labeling of individual sound events presented in the audio clip (for event detection tasks, the onsets and offsets of each event are also required); for separation tasks, the reference groundtruths are isolated tracks of each sound source. In general, it requires considerable human efforts to identify the sound events in each clip or to obtain the noise-free reference source tracks from mixtures; meanwhile, as audio data are unceasingly produced in the wild, sound clips without annotations are relatively easy to obtain.

Without reliance on annotations, self-supervised learning (SSL) methods have shown huge potential to learn low-dimensional representations that can generalize well to downstream tasks. The representations are learned via pretext tasks such as colorization [40], surrogate class prediction [41], masked token prediction [42, 43, 44], and instance discrimination [45, 46, 47]. While a majority of these recent advancements are made in computer vision or natural language processing, there are also a few successful applications of SSL methods in audio processing [48, 49, 50, 51, 52]. Additionally, these SSL approaches require a substantial amount of data to train, but it is not always realistic to train the models with all data at once. Considering this limitation, we propose an SSL framework [53] to learn audio representations in the context of continual learning in Section 4.2.

Besides self-supervised learning, semi-supervised learning is an effective alternative for handling data without annotations. In semi-supervised learning, the model is trained with some level of

supervision, where a portion of the training data comes with annotations and the rest does not [54]. Since unlabeled data is much easier to obtain, the scale of the unlabeled data is usually significantly larger than labeled ones in the majority of the prior studies in semi-supervised learning. While semi-supervised learning has been investigated for image [55, 56, 57] and natural language [58] tasks, it also gains popularity for audio applications such as sound event classification [59, 60] and automatic speech recognition [61]. In Chapter 5, we focus on semi-supervised learning for handling the data bottleneck for audio-text cross-modal representation learning [62] (Section 5.1), singing voice separation [63] (Section 5.2), and target speaker extraction [64] (Section 5.3).

CHAPTER 2: BACKGROUND

In this section, we introduce the background for audio classification and separation tasks and methods as well as the machine learning frameworks that serve as the foundation of the thesis.

2.1 AUDIO CLASSIFICATION TASKS

2.1.1 Audio Tagging

As shown in Figure 2.1 (Left), given an audio signal, we define audio tagging as the task of assigning the entire clip with a label. This task can be formulated as a multi-class classification problem $\mathbf{y} = f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^T$ is the recording signal represented as a vector, f is the classifier model, and $\mathbf{y} \in \mathbb{R}^C$ is the model's output indicating the likelihood of belonging to each of the C possible classes. Acoustic scene classification (ASC) [65] is one of the most popular applications of audio tagging, where the goal is to identify the background environment in which the recording is produced. Environmental sound classification (ESC) [66] is another emerging topic to recognize natural, urban, or daily sound classes from the recording. Besides identifying environmental sound, audio tagging is widely involved in speech processing tasks such as speaker identification [23] and emotion recognition [20, 21], and it is also applied to analyze music signals by categorizing musical genres [18, 19]. While a multitude of these aforementioned tasks assumes a single label for each clip, audio tagging can also be formulated as a multi-label classification where the input signal is associated with one or more labels. For instance, in-the-wild recordings are likely to contain multiple sound classes simultaneously, and a single speech clip can be characterized by more than one emotion. Under these scenarios, the classifier has to identify all relevant output classes.

2.1.2 Sound Event Detection

In addition to identifying the sound classes, sound event detection (SED) also determines the temporal boundary of the individual events. In contrast to audio tagging where a single output is made for the entire input sequence, SED systems predict the step-wise probabilities of the event classes at each time frame, as depicted in Figure 2.1 (Right). This task can be expressed in the form of $\mathbf{y} = f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^T$ is the input with T frames, $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \dots \mathbf{y}_T]^\top$ is the frame-wise output, and each output $\mathbf{y}_t \in \mathbb{R}^C, t = 1, 2, \dots, T$ is the estimated probability for each of the C classes. Depending on applications, SED systems can be further divided into monophonic or polyphonic, where the former assumes no overlapping events and the latter allows the presence of simultaneously occurring sound classes, which applies to more realistic scenarios. SED is broadly applicable in multiple settings, such as recognizing environmental or domestic events in smart home systems or wearable devices, determining anomalies of machines in factories, and identifying speakers' activity

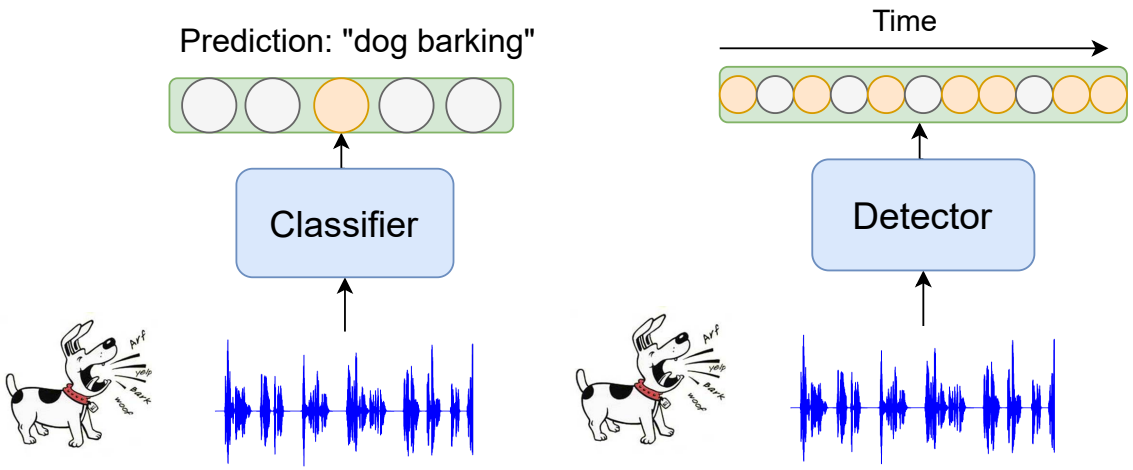

Figure 2.1: **(Left) Audio tagging:** The model makes one prediction for the entire input. **(Right) Sound event detection:** The model makes frame-wise prediction for the presence of event class.

in meeting scenarios. Furthermore, SED classifiers carry information that can benefit other audio processing tasks, such as speech enhancement [25, 67] and singing voice separation [63].

2.1.3 Methods for Audio Classification

Traditional approaches for sound classification tasks involve a two-step pipeline. In the first step, audio features are extracted from the time domain signal. Some popular representations of audio features include the magnitude spectrogram of Short-time Fourier Transform (STFT) and Mel-frequency cepstral coefficients (MFCCs). Additional feature transformation is optionally applied to reduce the dimensionality of the features with algorithms such as principal component analysis (PCA), independent component analysis (ICA), or latent Dirichlet allocation (LDA). The second step is to learn a classification model using the extracted features as input. The choice of classifier can be generative models including hidden Markov model (HMM) [68] and Gaussian Mixture Models (GMM) [69], or discriminative models such as support vector machine (SVM) [70, 71], k-nearest neighbor (KNN) [72] and multilayer perceptron (MLP) [73].

With the recent advances in deep learning, neural networks have obtained significant performance gains and outperformed traditional approaches in audio classification tasks. With successful application in image classification, convolutional neural networks (CNNs) are adopted to classify audio input. In [66, 74, 75], the spectrogram representation of the audio signal is treated as an image, and the input is processed by a sequence of two-dimensional convolutional layers before predicting the class label. The strong representation power of CNNs also enables end-to-end learning in which a single model performs feature extraction as well as classification using time domain signal as input [15, 76]. Instead of being manually designed, the features are learned jointly with the classification model and are optimized for specific tasks.

Different from image applications, the temporal structure of audio may provide cues for audio classification. Recurrent neural networks (RNNs) are proposed to model temporal dependency and capture the longer context in sequential input. Promising results of applying RNNs in speech activity detection are shown in [77, 78]. Recurrent layers are also used on top of the convolutional layers and show enhanced performance on urban sound classification [8, 12, 79, 80] and rare event detection [81]. Attention mechanism [6] is proposed as an alternative to RNNs for capturing temporal information and focusing on semantically relevant time frames in environmental sound classification [82], and it is shown to be effective when applied jointly with CNN and RNN layers [83].

2.2 AUDIO SOURCE SEPARATION

2.2.1 Audio Source Separation Tasks

Real-world audio signal often contains multiple events that potentially overlap with each other. In order to better analyze the signal, audio source separation is the task of isolating individual sound components from the mixture of sound sources. This problem can be formulated as

$$x(t) = \sum_{k=1}^{C} s_k(t), \tag{2.1}$$

where $x(t)$ is the input mixture, $s_k(t)$ denotes the source signal, and C is the number of sources. We would like to estimate each individual source component $\hat{s}_k(t) \approx s_k(t)$.

Depending on the nature of the source signals, audio source separation involves several domain-specific applications including but not limited to speech separation [84] (where the mixture only contains speech signals from multiple speakers), music separation [85] (where the input consists of musical instruments or singing voice), and universal source separation [86] (where any environmental event class may present in the mixture). Audio source separation can also be categorized according to the number of input channels. In multi-channel source separation, the input signal is collected from a microphone array, and the spatial cues from different channels can be used to infer the individual sources [87]. In single-channel (monaural) source separation, only one input channel is provided and we need to extract one or more sources from this single-channel input mixture. This task is challenging in that it requires solving an underdetermined system with only one equation and multiple unknowns. For the scope of this thesis, we focus on single-channel audio source separation tasks.

Speech enhancement [88] is a special case of audio source separation, where instead of separating all the sources, the goal is to extract only the speech and to suppress the noise. In this setting, we

are given a reverberant noisy mixture $x(t)$ as

$$x(t) = s(t) \circledast h(t) + n(t), \tag{2.2}$$

where $s(t)$ denotes the clean speech signal, $h(t)$ is the impulse response, $\circledast$ refers to the convolution operation, and $n(t)$ represents the ambience noise. The goal of speech enhancement is to estimate the speech signal $\hat{s}(t) \approx s(t)$ with an enhancement model. While A majority of studies on speech enhancement focus on a single-talker scenario, real-world recordings often involve conversations with more than one speaker, and the utterances potentially overlap. In this multi-talker scenario, the task of personalized speech enhancement (target speaker extraction) [89] becomes relevant when we are interested in tracking speech from only one of the speakers and treating the others as interference. The speech signal can be written as $s(t) = s_p(t) + s_i(t)$, where $s_p(t)$ is the speech signal from the primary or target speaker, and $s_i(t)$ is the signal from all interference speakers. To this end, the goal of personalized speech enhancement is to recover the target speech $s_p(t)$ from the input mixture and to suppress both the interference speech and background noise. By cleaning up the speech from the noisy mixture, speech enhancement can aid the processing of downstream applications such as automatic speech recognition (ASR), speaker identification, and speaker diarization.

2.2.2 Methods for Audio Source Separation

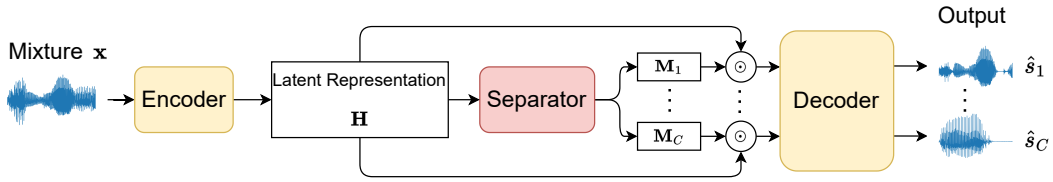

Figure 2.2: Masking-based separation framework based on deep learning. The encoder transforms the time-domain mixture signal to the latent representation, and the decoder converts it back to the waveform. Both modules can be predefined (e.g., STFT or inverse STFT) or learned (e.g., with one-dimensional kernels for convolution). The separator module estimates the mask for each of the C sources, which is applied to the latent representation of the mixture with element-wise multiplication.

Traditional approaches for source separation are based on generative models, including non-negative matrix factorization (NMF) [90, 91], independent component analysis (ICA) [92, 93], and hidden Markov model (HMM) [94, 95]. Under the maximum likelihood estimation (MLE) framework, these methods do not require clean reference signal for training, and the learned models are interpretable; however, they suffer from assumptions that impose limitations on the data distribution as well as the lack of tractable solutions.

With strong expressive power, neural-based source separation algorithms have surpassed tradi-

tional approaches on various source separation benchmarks. While unsupervised source separation using deep learning has garnered recent attention [87, 96, 97], the majority of neural-based source separation methods still follow the supervised learning framework, where the reference signals of the isolated sound sources are provided to the model. Instead of directly estimating the source components, the mainstream approach is to map the mixture signal into a latent space and compute a mask for each source with the neural network that can be applied to the latent representation. This masking-based separation framework usually contains three modules: the encoder f_e, the separator f_s, and the decoder f_d. The separation pipeline can be formulated as follows:

1. For the input mixture $\mathbf{x} \in \mathbb{R}^T$, compute the latent representation using the encoder, $\mathbf{H} = f_e(\mathbf{x}) \in \mathbb{R}^{F \times L}$, where F is the feature dimension and L is the temporal dimension;

2. For each source k, estimate the mask $\mathbf{M}_k = f_s(\mathbf{H})_k \in \mathbb{R}^{F \times L}$, and obtain the estimated latent representation $\hat{\mathbf{S}}_k = \mathbf{M}_k \odot \mathbf{H}$, where $\odot$ denotes element-wise multiplication;

3. Convert $\hat{\mathbf{S}}_k$ back to the time-domain signal with the decoder to obtain the estimated source $\hat{s}_k = f_d(\hat{\mathbf{S}}_k)$.

This procedure is illustrated in Figure 2.2.

Depending on the latent representation, these neural-based separation algorithms can be categorized as time-frequency (STFT) domain methods and time-domain (waveform) methods. For time-frequency domain methods, the encoder f_e and the decoder f_d are fixed to be the STFT and the inverse STFT transforms, respectively. The estimated masks hence represent the gain ratio for each time-frequency bin. The masks can be multiplied with the magnitude STFT spectrogram in conjunction with the mixture's phase [98], or they can be applied to the complex spectrogram with the phase-aware pipeline [99, 100]. The separator architecture can be based on fully-connected (FC) layers [99], convolutional layers [98, 101], or recurrent layers [102, 103].

For time-domain methods, instead of using a fixed latent representation, the encoder and the decoder are learned. A common design for the encoder module [26, 27, 104, 105] is by applying a one-dimensional convolution followed by a non-linearity transfer function, with

$$\mathbf{H} = f_e(\mathbf{x}) = \mathrm{ReLU}(\mathrm{Conv1D}(\mathbf{x})), \tag{2.3}$$

where ReLU denotes the rectified linear unit and applies element-wise to the output of the convolution with $\mathrm{ReLU}(x) = x \cdot \mathbb{1}[x \geq 0]$. To mirror the encoder, the decoder is a one-dimensional transposed convolutional layer:

$$\hat{s}_k = f_d(\hat{\mathbf{S}}_k) = \mathrm{ConvTr1D}(\hat{\mathbf{S}}_k). \tag{2.4}$$

There are a variety of choices for the separator architecture. The separator module of ConvTasNet [26] is based on stacks of convolutional blocks, which consist of one-dimensional convolution, non-linearity, and normalization; residual connections are applied to allow the gradients to flow

more directly through the network. The SuDoRM-RF [27] applies temporal downsampling and upsampling to rearrange the convolutional blocks into a U-shape architecture to enable information extraction at multiple resolutions for more efficient processing. The dual-path RNN [104] uses RNN layers to alternate the inter- and intra-chunk processing for improved modeling of both local and global dependency. The Sepformer [105] replaces the RNN layers with Transformer blocks [7] to obtain performance gain.

When the reference signal is available, the model training can be performed by comparing the estimated sources with the reference. For time-frequency domain methods, norm-based loss ($l1$ or $l2$) can be applied to the Fourier domain output. For time-domain approaches, the (negative) scale-invariant signal-to-distortion ratio (SI-SDR) [106] has been empirically shown to be an effective objective function, which is defined as follows:

$$\mathcal{L}_{\text{SI-SDR}}(\hat{\mathbf{s}}_k, \mathbf{s}_k) = -10 \log_{10} \frac{\| \frac{\hat{\mathbf{s}}_k^\top \mathbf{s}_k}{\|\mathbf{s}_k\|^2} \mathbf{s}_k \|^2}{\| \frac{\hat{\mathbf{s}}_k^\top \mathbf{s}_k}{\|\mathbf{s}_k\|^2} \mathbf{s}_k - \hat{\mathbf{s}}_k \|^2}, \tag{2.5}$$

where $\mathbf{s}_k, \hat{\mathbf{s}}_k \in \mathbb{R}^T$ are the reference and estimated signals, respectively.

When the ordering of the sources is meaningful (e.g, for personalized speech enhancement, musical source separation with known instruments), the model can be trained by directly minimizing the loss between each estimated signal and its corresponding reference and average across all sources:

$$\mathcal{L}_{\text{order}}(\hat{\mathbf{s}}, \mathbf{s}) = \frac{1}{C} \sum_{k=1}^{C} \mathcal{L}(\hat{\mathbf{s}}_k, \mathbf{s}_k), \tag{2.6}$$

where $\mathbf{s}, \hat{\mathbf{s}} \in \mathbb{R}^{C \times T}$ contain C reference and predicted sources and $\mathcal{L}$ is the signal-level loss such as the norm-based loss or SI-SDR. For tasks in which the ordering of the sources can be arbitrary (e.g, speech separation, universal sound separation), we need to consider all possible permutations of the sources during loss computation. The permutation-invariant training (PIT) [107] is proposed to address this challenge of permutation ambiguity, where the overall loss function is defined as

$$\mathcal{L}_{\text{PIT}}(\hat{\mathbf{s}}, \mathbf{s}) = \frac{1}{C} \min_{\pi \in \Pi} \sum_{k=1}^{C} \mathcal{L}(\hat{\mathbf{s}}_{\pi_k}, \mathbf{s}_k), \tag{2.7}$$

where Π is the set of all possible permutations of the C sources.

2.3 SELF-SUPERVISED LEARNING

Traditional supervised learning methods require a massive amount of labeled data to train. Additionally, representations learned from supervised frameworks often suffer from generalization errors. Compared to traditional supervised learning, self-supervised learning (SSL) approaches do not rely on annotations of data, and the learned representations are more robust and generalize

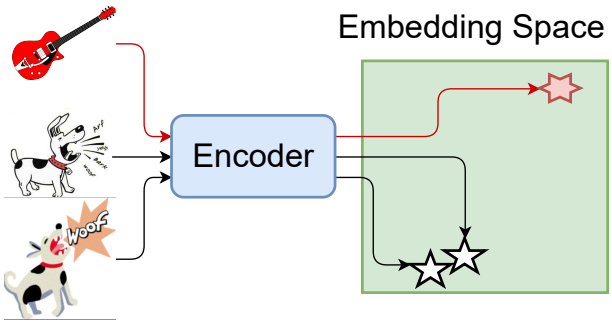

Figure 2.3: Similarity-based self-supervised learning. Similar input samples (e.g., dog barking) should lie close to each other in the learned latent space and apart from the embeddings of less relevant input (e.g., guitar).

better.

An SSL approach contains two major components: pretext tasks and learning objectives. The commonality of these pretext tasks is that pseudo labels can be generated from the input data itself, such as image inpainting [108], colorization [40], masked language modeling [42], and instance discrimination [46]. Based on the learning objective, SSL approaches can be classified as generative or discriminative frameworks. Generative models, such as Generative Adversarial Networks (GANs) [109] and Variational Autoencoders (VAEs) [110], aim to learn the distribution of the data itself, and the primary goal is to recover partial or full data distribution with the learned models. Discriminative approaches, on the other hand, estimate the decision boundary of the discrete pseudo labels for the pretext tasks.

Recently, similarity-based self-supervised learning [45, 46, 47] is becoming increasingly popular. Instead of training an end-to-end model (i.e., a model that directly predicts the label from the given input), an encoder network is trained to produce embeddings that are close to each other in the latent space for similar input, thereby capturing the semantics of similar instances. As shown in the example in Figure 2.3, the embeddings of two instances of the dog barking should lie much closer to each other than the embedding of the sound of the guitar. These pre-trained embeddings can be applied directly or fine-tuned with an output network on downstream tasks with few labeled data. The positive pairs (similar input samples) are obtained by applying data augmentations on the same input. This group of methods, however, tends to suffer from representation collapse, where the learned embeddings are nearly constant regardless of the input. To combat this issue, a straightforward method is to incorporate negative samples during training, introducing a discriminative pretext task of identifying the positive pair from the negative ones. For each sample in a batch of size N, SimCLR [46] augments this sample with random perturbations to obtain its positive pair, while the rest $2(N-1)$ samples from the same batch serve as negative pairs. On the other hand, MoCo [45, 111] samples negative pairs from a memory buffer of embeddings from the previous batches. The loss function for each positive pair of embeddings $\mathbf{z}_i, \mathbf{z}_j \in \mathbb{R}^D$ is the

normalized temperature-scaled cross-entropy loss (NT-Xent) [112]:

$$\mathcal{L}_{\text{NT-Xent}}(\mathbf{z}_i, \mathbf{z}_j) = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}[k \neq i] \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \tag{2.8}$$

where τ denotes the temperature parameter, and $\text{sim}(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^\top \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|}$ is the cosine similarity. While these approaches show promising results for several image benchmarks, they usually require large batch sizes for improved performances and therefore impose high requirements on memory. To enable memory-efficient training, Barlow Twins [113] introduces a redundancy loss by minimizing the correlation between different dimensions of the learned embeddings, which yields comparable performance to the aforementioned methods with significantly lower batch sizes. Alternatively, there are several approaches that can bypass sampling negative pairs while avoiding representation collapse. For instance, SwAV [47] performs clustering on the embeddings and enforces consistency of cluster assignments between positive input pairs. BYOL [114] and SimSiam [115] empirically shows that it is viable to learn non-trivial representations by minimizing the distance between positive pairs using the stop-gradient operation without having negative examples.

2.4 SEMI-SUPERVISED LEARNING

An alternative solution to make use of unannotated data is via semi-supervised learning, which is midway between supervised and unsupervised learning. As defined in [54], in semi-supervised learning setup, the training data $\mathcal{D}$ consists of a small supervised portion $\mathcal{D}_s = \{(x_i, y_i)\}_{i=1}^N$ with N input-label pairs and a large unsupervised portion $\mathcal{D}_u = \{x_j\}_{j=1}^M$ containing M input without annotation, where $N \ll M$.

While the limited labeled data $\mathcal{D}_s$ can be leveraged under the conventional supervised training framework, there are two major schools of methods for exploiting the massive amount of unlabeled data [116]. The first approach is consistency regularization [55], which introduces a separate learning task for $\mathcal{D}_u$. It assumes that the model should produce similar output for the same input under reasonable perturbations. This concept is closely related to the similarity-based self-supervised learning, but instead of having a pretext task for classifying positive and negative examples, the learning is achieved with a loss term that minimizes the differences between the output of the original and perturbed input.

The second group of methods is based on bootstrapping [117], where a pre-trained model is used to assign pseudo-labels for $\mathcal{D}_u$ to provide additional input-label pairs for training. One such method is the self-training [57] depicted in Figure 2.4. The pipeline of self-training can be summarized as follows: The first step is to train a teacher model $f^{(0)}$

which repeats the procedure of

1. Train a teacher model $f^{(0)}$ using the labeled dataset $\mathcal{D}_s$;
2. Assign pseudo-labels for data from $\mathcal{D}_u$ with f_0; denote the output dataset as $\mathcal{D}_u^{(1)}$;

3. Training a student model $f^{(1)}$ using both $\mathcal{D}_s$ and $\mathcal{D}_u^{(1)}$;

4. Optionally, making $f^{(1)}$ the new teacher model, and repeating steps 2 and 3.

To summarize, consistency regularization introduces an unsupervised learning task for the unlabeled data, whereas bootstrapping provides labels to the unlabeled data so that it can fit under the conventional supervised learning framework.

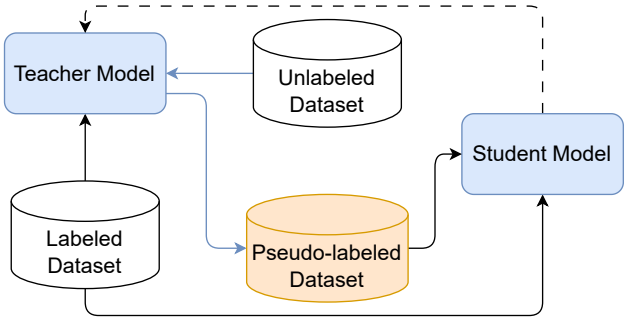

Figure 2.4: Self-training with bootstrapping. Paths with solid black lines indicate the data is used for training the network. The path with solid blue lines stands for the process of assigning pseudo-labels with the teacher model after being trained with labeled data. The black dashed line refers to the step where the student model becomes the new teacher in the next iteration.

2.5 CONTINUAL LEARNING

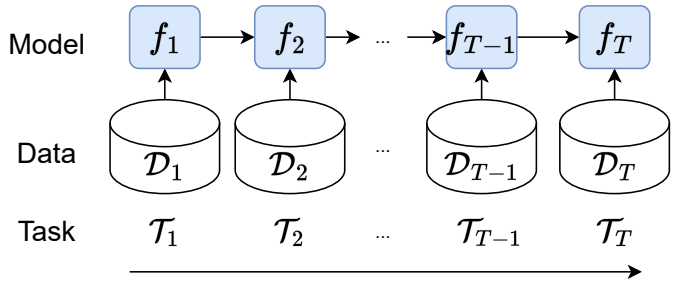

Figure 2.5: Pipeline of continual learning. Given a sequence of T tasks, the model $f_t (1 \leq t \leq T)$ is sequentially updated on the tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ with the associated datasets $\mathcal{D}_1, \ldots, \mathcal{D}_T$ that potentially come from different distributions.

Besides requiring labeled data for training, supervised training also assumes that data samples are independent and identically distributed (IID). Conventionally, this is achieved by sampling batches of data from a large fixed dataset for each optimization step, and the process repeats until the model converges. In practice, however, it might not be reasonable to assume that the full dataset is available at once. Instead, the dataset is not necessarily stationary, as new data samples may arrive throughout the model training. As illustrated in Figure 2.5, continual learning

(CL), or incremental learning, describes the procedure in which the model is trained on a sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T$ and the model's weights are continually updated. Each of these tasks is associated with its own dataset, $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T$, and these datasets are not necessarily drawn from an identical distribution. The model is trained on these datasets one at a time with limited access to past or future data. At the end of task t, the model f_t should not only capture information from the current dataset $\mathcal{D}_t$, but also maintain good performance on all previous tasks, $\mathcal{D}_1, \ldots, \mathcal{D}_{t-1}$, in which it has been trained on. For supervised classification tasks, the dataset $\mathcal{D}_t$ comes with pairs of input and labels $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n_t}, (\mathbf{x}_i, y_i) \in \mathbb{R}^D \times \mathcal{Y}_t$, where $\mathcal{Y}_t$ is the set of class labels in $\mathcal{D}_t$.

Depending on whether the task identity is provided at inference, task-aware and task-agnostic CL are introduced in [118]. In general, task-aware CL requires the task label as a conditioning input and involves separate prediction heads for each task, whereas task-agnostic CL shares the prediction network without information on task identity. Task-agnostic CL could be further categorized as domain-incremental and class-incremental settings. For any two distinct tasks $\mathcal{T}_i, \mathcal{T}_j, i \neq j$, under the setting of domain-incremental learning, the associated datasets $\mathcal{D}_i, \mathcal{D}_j$ do not share any data in common (i.e., $\mathcal{D}_t \cap \mathcal{D}_{t'} = \emptyset$), and $\mathcal{D}_t, \mathcal{D}_{t'}$ are drawn from distinct distributions. In class-incremental learning, each task is associated with a distinct set of class labels, where $\mathcal{Y}_t \cap \mathcal{Y}_{t'} = \emptyset$ for $t \neq t'$.

Learning models that can capture new information while preserving prior knowledge under continual learning settings is a challenging task. A seemingly straightforward solution is to finetune or retrain the model from scratch with both old and new data; however, such approaches are not always feasible due to the lack of access to old data samples or the prohibitive cost of time and computational resources. On the other extreme of the spectrum, naively training on a sequence of datasets is also a suboptimal solution: this method would result in *catastrophic forgetting* [32], a phenomenon where the performance on previous tasks is impaired as the model gets trained progressively on new tasks. A plethora of continual learning (CL) approaches have been proposed to learn new tasks efficiently while retaining the performance of prior tasks without having full access to the previous datasets. These CL methods can be divided into regularization-based, model-based, and replay-based approaches.

When the model is trained on the new task, regularization-based methods encourage the model to behave similarly to its previous snapshot. These methods usually require storing a copy of a model at the end of each task. This is achieved in EWC [33]by penalizing changes of certain parameters that are important to previous tasks. In the learning without forgetting (LwF) [34], the network's outputs are encouraged to match the outputs from the model obtained at previous tasks using a distillation loss. With regularization, models are able to reduce the effects of catastrophic forgetting without revisiting any data from the past. Nevertheless, regularization-based methods still suffer from forgetting when trained with a large number of tasks.

Model-based approaches assign task-specific parameters for each additional task. The PackNet [37] leverages network pruning, identifying weights important to previous tasks while freeing up the rest for the learning of new data. The Piggyback [38] extends the PackNet by jointly learning task-specific masks that can be applied on a shared backbone network, enabling the processing of

multiple sequentially learned tasks on a single network. Both Progressive Neural Network [119] and Expert Gate [120] replicate the backbone network for each additional task, while lateral connections between networks are used in the former and a gating mechanism is used to select task-specific for the latter. While avoiding forgetting without relying on storing data, these approaches usually involve storage overhead for saving weights of the task-specific parameters, and the overhead increases linearly with the number of tasks.

Replay-based methods require retaining data from the past, and they usually achieve superior performance to replay-free approaches. At the end of each task, iCaRL [35] maintains a fixed size exemplar set by computing each sample's distance to the class-wise mean representation, storing the nearest prototypes, and discarding the rest. GEM [121] and AGEM [122] store a random subset into the buffer and apply constraints on the similarity of the gradients obtained on the current data and buffered samples. Inspired by [121, 122], a gradient-based sample selection algorithm is proposed in [123] to replace the constrained optimization. Despite the strong performance, replay-based methods using real data from the past are not always infeasible in practice since access to previously seen data may be limited due to storage constraints or privacy regulations. The generative replay technique [36] is proposed to overcome this limitation, where a generative network is trained in parallel to the classifier. For each task, both the classifier and the generator are trained with the current data along with samples produced by the generator trained on previous tasks. By simulating the replay buffer with a generator, this approach avoids storing actual data while achieving strong performance on continual classification tasks.

CHAPTER 3: ARCHITECTURE DESIGN FOR SOUND EVENT DETECTION

Audio signals are presented in the form of time series, and sound events in our daily life are usually characterized by distinct temporal structures. For the task of sound event detection, analyzing neighboring time frames may provide cues for understanding the sound events at the current time step. For instance, suppose we know the previous time frame is the onset of a baby crying, then the event is very likely to occur at the current step, given that the sound of the baby crying usually lasts for a few seconds. As described in Section 2.1, recurrent neural networks (RNNs) are known for their effectiveness in modeling sequential information and have achieved promising results for detecting sound events [77, 78]. With a hidden state vector, RNNs are able to aggregate and propagate information back and forth.

Conventional RNNs for natural language processing take one-dimensional signals (i.e., text tokens) as input, and the hidden state vector gets unrolled across the temporal dimension. Audio data, on the other hand, may come with extra dimensions. A common approach to model audio signals with neural networks is by first transforming the one-dimensional time-domain signal into two-dimensional time-frequency domain representations (e.g., STFT spectrograms). Another example is spatial recordings obtained from microphone arrays, where the additional dimension is related to the input channels.

In this chapter, we propose two variants of the RNNs that can adapt to the processing of multi-dimensional audio signals. Meanwhile, these novel architecture designs effectively address the challenges of learning under constraints for data collection. Section 3.1 presents the Multi-View Networks (MVNs) for multi-channel speech event detection. Unrolling across both the channel and the time dimensions, the MVNs do not require collecting a comprehensive set of multi-channel audio data to train, and it is able to generalize well under an unseen number of channels during evaluation. Next, we integrate multi-dimensional RNNs with adaptive computation in Section 3.2 for single-channel sound event detection. At each time step, the model intelligently decides the number of sub-bands it needs to process in order to make a confident prediction about the sound class. The proposed architecture improves the robustness of the model under scenarios where part of the information is missing from the data.

3.1 MULTI-CHANNEL SPEECH DETECTION WITH MULTI-VIEW NETWORKS

3.1.1 Multi-Channel Sound Event Detection

Sound event detection is becoming an increasingly relevant problem, one in which we are seeing a lot of activity and progress in the last few years. In this section, we focus on the scenario with a lot of acoustic sensors, but neither they all record a clean enough signal for the task at hand, nor

[1]Part of this chapter has been published in [29, 30].

that they are all recording at any time. For instance, consider the case of a few people in an office setting. Each person will likely have a cell phone with a couple of microphones, a laptop with a few more, and maybe some mic-enabled wearable devices; there might be a room microphone tethered to a conferencing system, or audio-enabled desktop computers in the room, perhaps a few hearing aids, and maybe some security microphones as well. In this situation, we might want to perform audio sensing tasks (e.g. diarization); although we have access to a large number of recordings in the room, not all will be of use. For instance, some cell phones might be surrounded by interfering noise and hence provide a non-informative signal, whereas others might be right next to the type of sound we want to detect. Our goal is to explore algorithms that will not be misled by channels with low-informational content and not be tethered to a fixed number of channels. In order to do so we introduce the concept of the *multi-view network* (MVN) for the purpose of sound event detection. The framework that we present here considers a simple classifier but is easily amenable to more elaborate extensions in order to facilitate state-of-the-art classification and detection models, as long as they fall under the umbrella of a deep learning model.

Deep learning models for monaural or binaural audio classification have been explored in many settings. Several deep architectures have shown to be powerful tools for the tagging task [124, 125, 126, 127]. Neural-based approaches for multi-channel audio classification have been investigated in automatic speech recognition, where the multi-channel input is first aggregated by a beamformer front end before passing into the acoustic model (both the beamformer and acoustic model are parameterized by feedforward networks or RNNs) [128, 129]. Multichannel acoustics are also used in modeling domestic sound events, where the state-of-the-art methods rely on source separation, dereverberation, and data augmentation as preprocessing steps and CNNs as back ends to classify multi-channel audio input [130, 131]. These approaches, however, only explore scenarios with a tight number of channels and have limited discussions on practical challenges associated with multi-channel data. When a deep audio classification model is trained to perform classification on, e.g., four channels it typically can't guarantee the ability to leverage information when more channels are provided or to function when some are missing. In contrast to neural-based methods, classic beamforming approaches can scale to an arbitrary number of input channels. However, with a fixed number of channels available, learning methods are typically superior to beamforming. Here we seek to remedy this by using network architectures that accept inputs of variable sizes, allowing us to train on a fixed number of channels and deploy on any other number of channels at inference time.

Our work extends the Multi-View Networks for denoising [28]. The denoising model attempted to predict clean spectra from noisy recordings, our classification model presented here extends that idea to that of performing classification. Our results show that MVNs are fit for classification and that they can handle channel disturbances in a dynamic environment with simulated Room Impulse Responses several times larger than our Short-Time Fourier Transform(STFT) frames.

3.1.2 Multi-View Networks

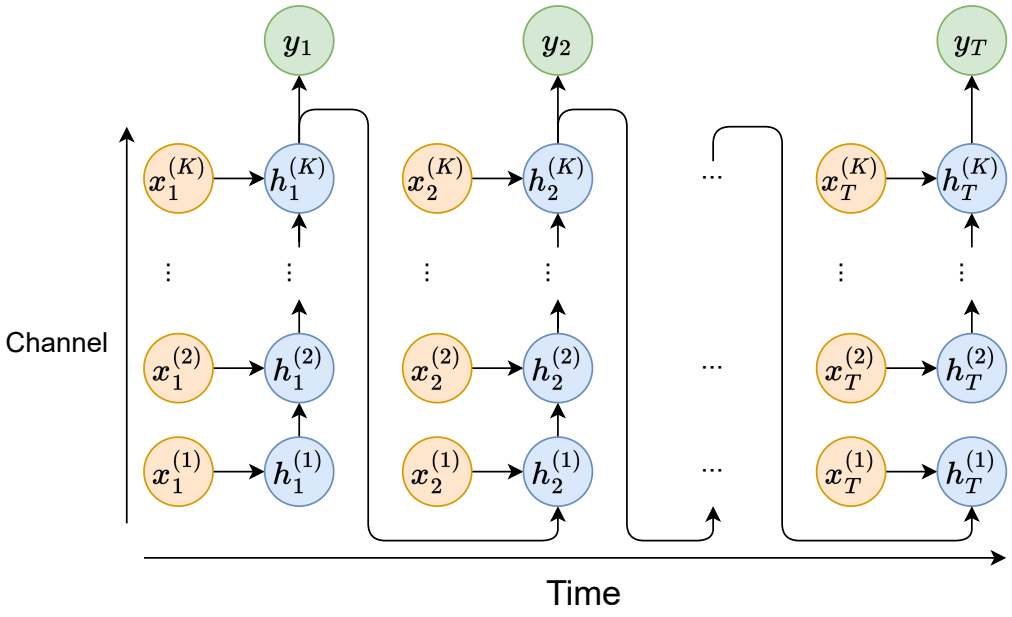

Figure 3.1: Forward pass for MVN. Channel-wise information is aggregated by observing all shared time steps of each channel before making a prediction for this frame. Then, the hidden state after observing the last channel feeds into the processing of the first channel of the next time step, enabling the propagation of temporal information.

RNNs have been commonly applied for single-channel audio classification or detection tasks. They typically operate on a chosen short-time spectral representation and unroll across time to leverage the temporal dependencies between successive spectral frames. The forward pass for RNNs is defined as

$$\begin{aligned} \mathbf{h}_t &= \sigma_h(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h \mathbf{h}_{t-1}), \\ \mathbf{y}_t &= \sigma_y(\mathbf{W}_y \mathbf{h}_t), \end{aligned} \tag{3.1}$$

where $\mathbf{x}_t \in \mathbb{R}^F$ is the $F-$dimensional spectral input at frame t, $\mathbf{W}_h, \mathbf{W}_y, \mathbf{U}_h$ are the learned weights, $\mathbf{h}_t$ is the hidden-state representation at time t, σ_h, σ_y are non-linearity functions for the hidden and output layers, respectively, and $\mathbf{y}_t$ is the prediction at time t. Due to the ability to process arbitrary-length input, RNNs can be trained and tested by using sequences with different numbers of frames (i.e., training with shorter sequences and testing on longer ones).

In this work, we study multi-channel voice activity detection where both the number of training and test time frames and channels are potentially different. We propose the Multi-View Network (MVN) [28], a multi-dimensional extension of RNNs that can unroll across both the channel and the frequency dimension to enable the processing of arbitrary numbers in both channels and time frames. As shown in Figure 3.1, for each time step the model processes the information for all K

channels in a fixed order before proceeding to the next frame. The forward pass is defined as

$$\mathbf{h}_t^{(k)} = \begin{cases} \sigma_h(\mathbf{W}_h \mathbf{x}_t^{(k)} + \mathbf{U}_h \mathbf{h}_t^{(k-1)}) & \text{if } 1 < k \leq K \\ \sigma_h(\mathbf{W}_h \mathbf{x}_t^{(k)} + \mathbf{U}_h \mathbf{h}_{t-1}^{(K)}) & \text{if } k = 1 \end{cases}, \tag{3.2}$$
$$\mathbf{y}_t = \sigma_y(\mathbf{W}_y \mathbf{h}_t^{(K)}),$$

where $\mathbf{x}_t^{(k)}$ is the spectral frame t of the $k-$th input channel, $\mathbf{h}_t^{(k)}$ is the hidden state representation after observing the $k-$th channel of time t, and the other symbols follow the same definition as in Equation (3.1). Because this recurrence operates on both channel and time dimensions, it enables the model to process input with an unseen number of channels and time steps during inference.

3.1.3 Experimental Setup

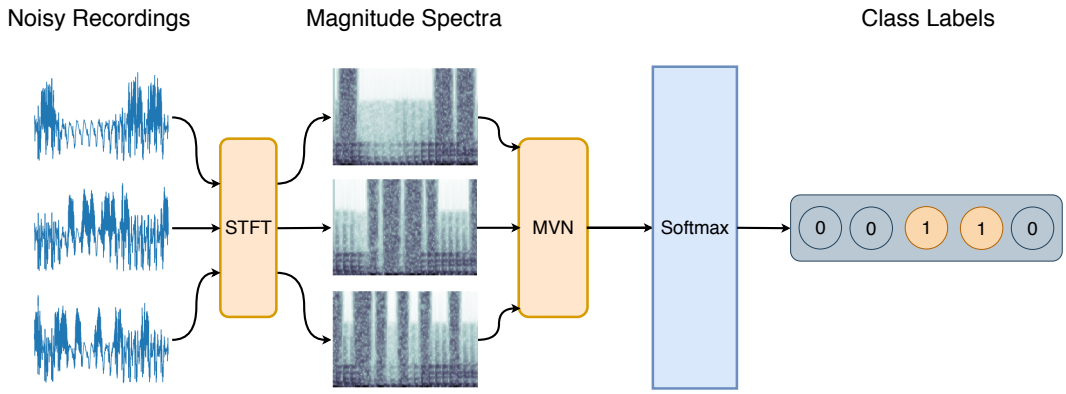

Figure 3.2: Audio event classification pipeline with an MVN. For a multi-channel mixture, we first take the magnitude spectra for each channel with STFT. The network takes the magnitude spectra as input and predicts if each frame contains speech.

We next introduce the setup of the audio classification experiments, including the data set, the baseline models, and two experiments evaluating the performance of the models. The models and the data correspond to a binary classification task in which each input is classified as either speech or not. Figure 3.2 illustrates the general pipeline for the experiment. These experiments model various scenarios with many microphones where most of the recorded signals are extremely noisy or contain little information. This reflects the potential application of this model in IoT sensing problems.

Dataset

We prepare the data set for the binary audio classification experiments by mixing speech and noise segments. The speech segments are selected from the TIMIT Corpus, which includes 25,200 recordings with 630 speakers of 8 major dialects of American English, each reading 10 phonetically

rich sentences [132]. The noise segments are selected from 13 different background noise recordings such as "Airport", "Babble" and "Restaurant" noises [133].

The dataset is made up of two-second noisy mixtures at a sampling rate of 16 kHz. To create a sequence, we randomly select a two-second segment from one of the noise recordings and a segment between zero to two seconds in length from one of the TIMIT speech recordings. Next, we normalize each of the noise and speech segments to unit variance. We then mix them by adding the speech segment to a random position within the two-second noise segment to obtain a single-channel mixture. The resulting mixtures are approximately half speech and half background noise. We apply STFT with a 1024 pt window and a 512 pt hop on each channel of the mixture and take the magnitude spectrogram as the input to the models.

Based on this single-channel mixture, we propose different ways to generate multi-channel mixtures. For our experiments, we set the per-channel SNR of a noisy mixture by scaling it to the desired SNR in decibels (dB). For the training and validation set, all mixtures contain four channels whose SNR values are evenly spaced between -5dB and 5dB. For the test set, the number of channels is ranged between 2 and 30, emulating scenes with different numbers of available sensors in some ad-hoc networks. We propose two scenarios for generating multi-channel mixtures for testing. In the first scenario, each additional channel has a lower SNR value than the prior channels. Specifically, for a mixture of $K \in \{2, 3, \ldots, 30\}$ channels, the SNR values for the channels are $0\text{dB}, -1\text{dB}, \ldots, -(K-1)\text{dB}$. The more channels, the lower the average SNR value. We call this scenario "decreasing SNR". We also propose an "increasing SNR" scenario in which a mixture of $K \in \{2, 3, \ldots, 30\}$ channels contains SNR values of $-29\text{dB}, -28\text{dB}, \ldots, (-29+K-1)\text{dB}$. The more channels, the higher the average SNR value. In both scenarios, the channel indices are randomly shuffled.

By providing the model with progressively worse channels we test the ability to ignore noisy information. By providing progressively better channels we test the ability to leverage new information. These setups expose the model to a diverse set of SNRS mimicking sensor networks in a chaotic environment.

Baseline Models

We now introduce three different binary classification strategies as the baseline for the experiments. Each model takes the mixture's magnitude STFT spectra as input, and then pass it into a GRU with 512 hidden units unrolling across time followed by a softmax layer that classifies each input frame as either noise or signal.

We first consider the **Averaging Input** model. This model averages the magnitude spectra across channels for each mixture, and it then feeds the averaged spectra into the network. The output of the softmax layer is the estimated probability distribution of each frame being noise or signal:

$$h_{\mathbf{\Theta}}(\mathbf{X_t}) = \text{argmax}_{c\in\{0,1\}} P(y_t = c|\overline{\mathbf{x_t}}; \mathbf{\Theta}) \tag{3.3}$$

$$\overline{\mathbf{x_t}} = \frac{\sum_{k=1}^{K} \mathbf{x_t^{(k)}}}{K} \tag{3.4}$$

where $\mathbf{X_t}$ is the set of magnitude spectra at time t for a given mixture with K channels, $\mathbf{x_t^{(k)}}$ is the spectral frame at time t for the k^{th} channel, $c \in \{0, 1\}$ is the binary label for each frame, and $\mathbf{\Theta}$ is the set of model parameters.

The second baseline is the **Averaging Output** model. This model takes the magnitude STFT spectrogram for each channel of the mixture and processes each channel independently. For a mixture with K channels, we will therefore have K different output probability distributions from the softmax layer. We obtain the probability for each frame by averaging the K softmax probability distributions:

$$h_{\mathbf{\Theta}}(\mathbf{X_t}) = \text{argmax}_{c \in \{0,1\}} \frac{\sum_{k=1}^{K} P(y_t = c | \mathbf{x_t^{(k)}}; \mathbf{\Theta})}{K}. \tag{3.5}$$

The third baseline is the **Max Output** model. Similar to the Averaging Output model, this model takes the magnitude spectrogram for each of the K channels of the mixture and produces K output distributions for each frame. Instead of averaging the output probabilities, we use the prediction with highest confidence:

$$h_{\mathbf{\Theta}}(\mathbf{X_t}) = \text{argmax}_{c \in \{0,1\}} \max_{k \in \{1 \ldots K\}} P(y_t = c | \mathbf{x_t^{(k)}}; \mathbf{\Theta}). \tag{3.6}$$

Training Setups

For all experiments, we train with 250 batches of 40 k-channel mixtures per epoch(10k mixtures total). We use Adam [134] with an initial learning rate of 1e-3 to minimize the cross-entropy loss, and we train each model for 100 epochs and use the model saved at the epoch with the lowest validation loss. The learning rate is decreased by a factor of 0.25 every 20 epochs.

3.1.4 Results and Discussions

We now report the performance of the MVN and the baseline models in two experiments.

Anehoic Mixtures

In the first experiment, we create multi-channel input data with instantaneous mixtures of speech and environmental noise data according to Section 3.1.3. Since there is no reverberation applied during the synthesis process, we refer to this setup as "Simple Mixtures". Besides shuffling the channel indices, we also shuffle the SNR for each channel at each time step in the time-frequency domain. With such a setup, we aim to model a dynamic environment in which some sensors move around the signal of interest or the signal received by the sensors is disturbed intermittently.

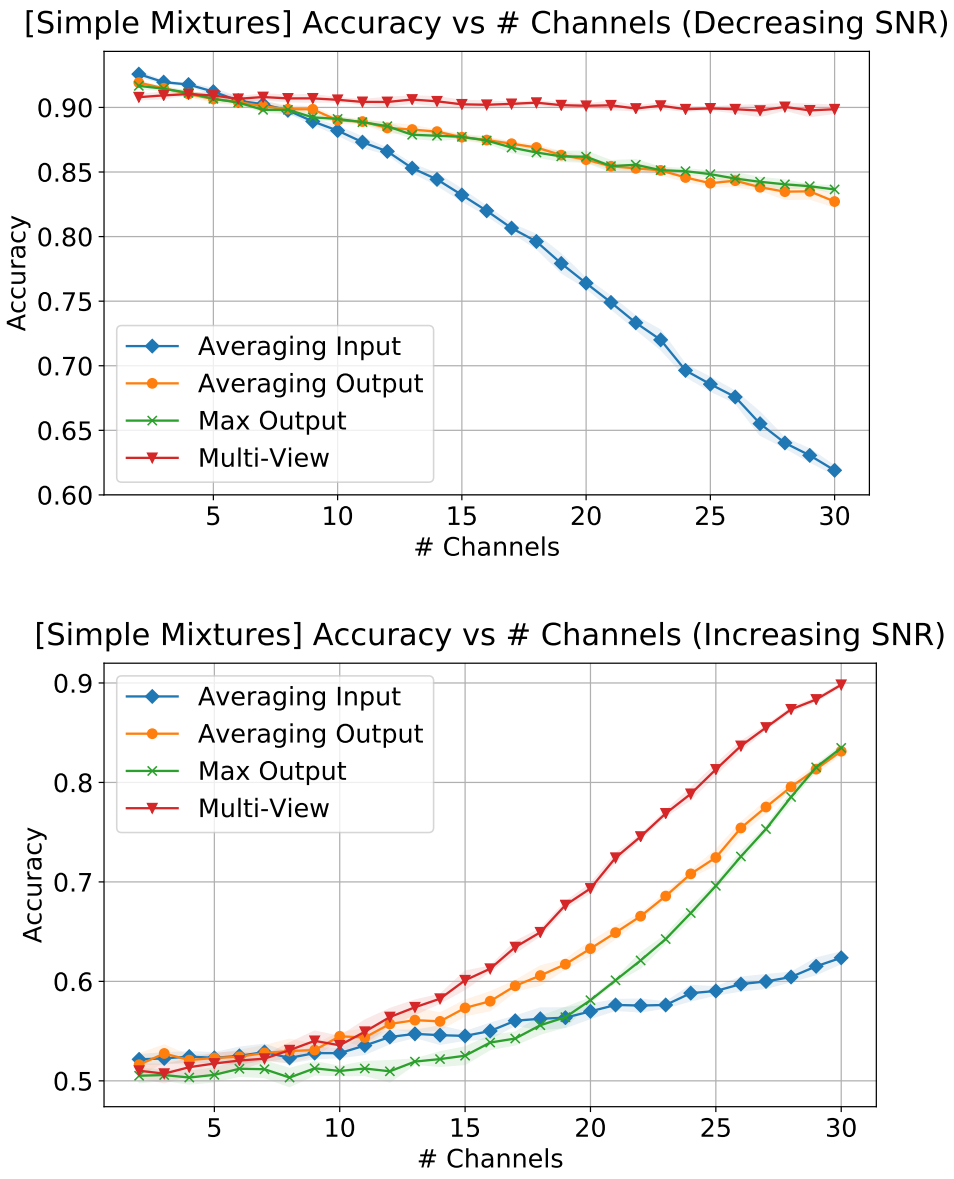

Figure 3.3: Test accuracy of speech activity detection using anechoic mixtures with the decreasing (top) and increasing (bottom) SNR scenario. Each experiment is repeated five times with different initialization. The markers and the shades represent the mean and standard deviation, respectively, for test accuracy.

Figure 3.3 (top) shows the prediction accuracy of the MVN and the baseline models from 2 to 30 channels with decreasing SNR. All models are trained on four-channel mixtures with SNRs uniformly spaced between -5 and 5dB. For decreasing SNR, the SNR value decreases by 1 dB for each additional channel. The models have similar performances when the number of channels is less than 10; however, as it goes beyond, the performance of the MVN is more stable than the baseline models. The result shows that the MVN is least affected by the channels with low SNR

values compared to the baseline models. For increasing SNR, as shown in Figure 3.3 (bottom), each extra channel is 1 dB higher than the previous channel. The MVN takes the least number of channels to achieve some desired accuracy, indicating that the MVN collects information more effectively than all the baseline models. Moreover, the MVN is able to generalize from training on a fixed number of channels and a limited range of SNRs to testing on a varied number of channels with a large range of SNRs.

Room Simulation

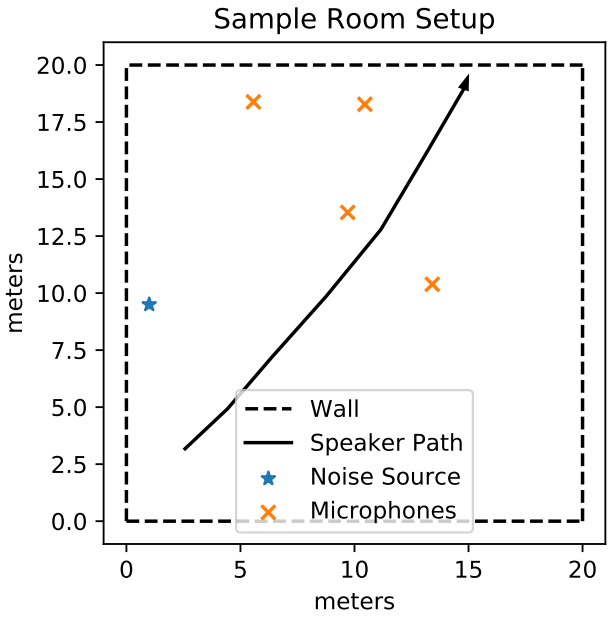

Figure 3.4: Sample setup with one noise source, four microphones, and a moving speech source in a 20-by-20 metered room. Diffuse noise is not pictured.

In this experiment, we use Pyroomacoustics [135] to model a room with length and width both being 20 meters using the image source model using fourth-order echoes. We train the model on many microphone-speaker-noise source geometries and test on unseen ones. To construct a simulated room we first simulate a noiseless moving speech source. Then, using the same microphone geometry we simulate a noise source randomly placed on a grid. To construct a mixture we take a speech recording and a noise recording which correspond to the same microphone geometry and mix them at some SNR. The separate simulation of noise and speech lets us mix and match to construct training and testing environments without having to explicitly simulate every combination. For training we use mixtures with per channel SNRs between -5 and 5 dB then at test time we experiment with both "decreasing SNR" and "increasing SNR" as described in Section 3.1.3. All speakers are modeled as point sources. In general, the simulated room impulse responses are five

times the length of an STFT window.

Inside the simulated rooms we place 2 to 30 microphones, a noise source and a diffuse noise. The speech source moves in a noisy linear path from one corner of the room to another. The indices of the microphones are randomized. Figure 3.4 shows one possible configuration of a simulated room.

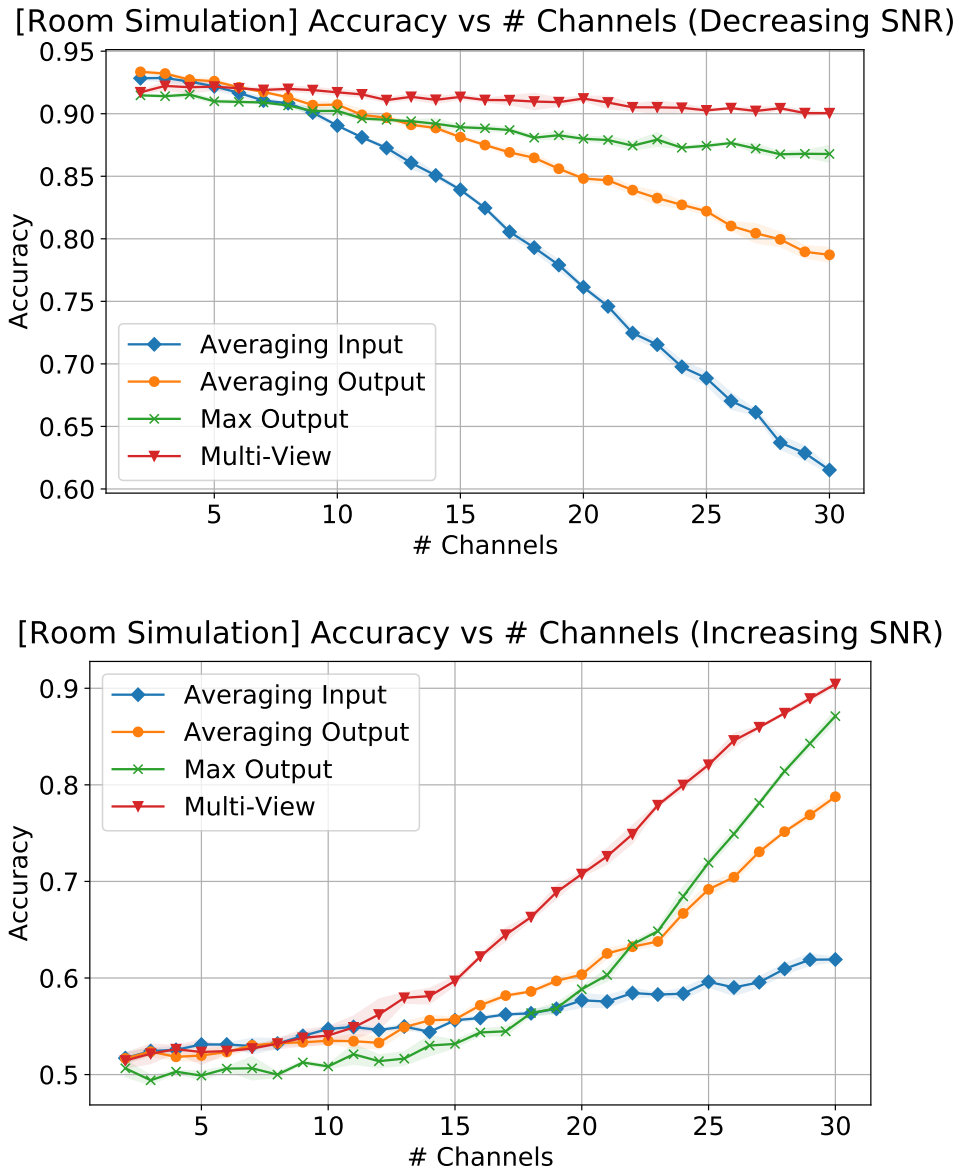

Figure 3.5: Test accuracy of speech activity detection using mixtures obtained from room simulation with the decreasing (top) and increasing (bottom) SNR scenario.

Figure 3.5 depicts the accuracy of the MVN and the baseline models for speech classification. The mixtures range from 2 to 30 channels. We observe a pattern similar to the experiment with the anechoic mixtures in Section 3.1.4. Among the four models, the MVN is least affected by noise

and room impulse responses, and it collects information most effectively from channels that may have high SNRs.

3.1.5 Conclusion

We have proposed a neural network method for multi-channel audio classification using an RNN that unrolls across both channels and time. We demonstrate that the proposed architecture can be deployed to unseen numbers of channels and unseen room geometries at test time. The system is robust to noisy channels in a highly dynamic environment, making it unnecessary to eliminate certain channels as a preprocessing step. Moreover, the system demonstrates the ability to leverage information effectively from a limited number of clean channels when they appear among many noisy channels. As such, this model is capable of being trained once and then deployed in settings with a dynamically changing number of sensors (e.g. in an IoT case), without requiring retraining or any modifications. The proposed framework is not limited to binary classification and can be used for multi-class or multi-label classification, but also with any other type of neural network architecture as long as the channel unrolling takes place. We hope that this will form the basis of powerful future systems that have to operate under uncertainty on the number of input channels, as opposed to resorting to simple averaging of voting schemes that are not as adept in taking the data into account.

3.2 SOUND EVENT DETECTION WITH ADAPTIVE FREQUENCY SELECTION

3.2.1 Efficient Methods for Sound Event Detection

Deep neural networks have achieved significant progress in a variety of audio processing tasks. Large models with millions of parameters have delivered new state-of-the-art results in sound event detection [8], scene classification [13], audio captioning [136], and more [17]. The convolutional recurrent neural network (CRNN) [8], composed of a series of convolutional and recurrent layers, has become a dominant architecture in event detection competitions [137]. The success of the CRNN architecture can be attributed to its ability to combine local information with convolutional layers and temporal information with recurrent layers. Several promising lines of work are based on improving these structures [9, 10, 11]. However, the deployment of these deep neural networks is complex as different devices have different computational, storage, or bandwidth limits.

To address this issue, multiple concurrent lines of work have proposed techniques to reduce the complexity of the model for audio processing tasks. These techniques include sparsity-based approaches [138, 139, 140, 141], and quantization based approaches [142, 143, 144]. In addition, a variety of weight-sharing approaches have demonstrated strong results. Sub-band processing, an approach that shares parameters across bands of input, has successfully decreased computational costs while maintaining strong performance in sound event detection [145], acoustic scene classi-

fication [13], speech recognition [146, 147], speech enhancement [148] and source separation tasks [149, 150]. However, these methods typically assume that models have access to a fixed number of bands or inputs. Such an assumption is not realistic in real-world applications where the test devices have limited bandwidth, or several bands are contaminated during transmission or storage.

In scenarios where the number of input channels or bands is time-varying, systems must scale to large or small numbers of inputs. The multi-view network (MVN), introduced in Section 3.1, provides a way to train a model on a fixed number of input channels and test on a varying number of inputs [28, 29]. Often, the number of inputs is determined by the computational budget. Therefore, adaptive computation models have been proposed to minimize computation in a time-varying fashion. These models also incorporate computational expense into their training objectives. Adaptive computation time (ACT), originally proposed by [151], achieves this by constructing weighted sums of hidden states and penalizing excessive hidden state usage. The ACT framework has been successfully applied to a variety of other tasks including early exiting of convolutional neural network (CNN) layers [152, 153], adaptive depth Transformers [154], microphone selection [155] and in visual question answering through the differentiable adaptive computation time (DACT) mechanism [156].

In this part of the thesis, we propose the *Hidden-state Interpolated Differentiable Adaptive Computation Time* (HIDACT), a variant of CRNN that is trained with a fixed computational budget and deployed with an intelligently adaptive budget. We demonstrate this technique in a monophonic sound event detection task where the proposed models vary the number of frequency bins requested across time. We compare these models to a suite of baselines, including full-band models and sub-band models observing all frequency sub-bands. To verify that the proposed models are compatible with the aforementioned advanced neural network architectures for event detection, we integrate the network design and learning objectives of the adaptive computation with the CRNN architecture in [8, 9, 10], as it is the most popular architecture in recent DCASE challenges [137]. We evaluate the classification performance with frame-wise macro F1. We also evaluate the computational cost of all techniques with the multiply-accumulate (MAC) operations [157]. Empirical results show that the proposed HIDACT can gracefully scale their computational budgets effectively, allowing a single model to be trained and then deployed on devices with a varying computational budget. Furthermore, during test time, HIDACT is able to generalize well under partially missing information, accurately predicting sound classes when only a limited number of frequency sub-bands are presented to the model.

3.2.2 Adaptive Computation with Neural Network

As a precursor for processing input sequences with a variable length, the MVN proposed in Section 3.1 is designed for processing an arbitrary number of input channels for multi-channel speech detection. Following the formulation in Equation (3.2), for each frame, the network observes information from all channels before making a prediction. Then, the hidden state of the last step

of the current frame is passed to the processing of the first feature of the next frame, allowing the propagation of temporal information. Despite being trained on a fixed number of channels, MVNs can adapt to a wide range of input channels during test time, even under cases where a significant proportion of the input channels contain little informative content.

MVNs are *dynamic* but not *adaptive*: the number of processing steps may change over time, but MVNs always make use of all available features (channels) at each frame regardless of the input and cannot determine the amount of information to process. In the context of single-channel sound event detection, the number of frequency sub-bands at each time step can be treated as the number of processing steps. To further reduce the amount of computation, it is desirable to have an *adaptive* system that intelligently selects the number of frequency sub-bands for each frame instead of processing all sub-bands. Consider an audio clip with a few seconds of conversation followed by music. Intuitively, the low frequency bins of the speech frames have richer content, whereas the high frequency bins of the music frames contain more information. For each frame, the model unrolls from the lowest frequency sub-band to the highest one. At a speech frame, the model with adaptive computation is able to detect speech after observing the first few sub-bands, and it can intelligently stop processing the current frame and continue to the next frame. This can be achieved with an implicit confidence model, which guides the model to halt the computation at the current step if it predicts that there is no significant performance gain for processing additional information. We next introduce formulations of adaptive computation with neural networks, which serve as the foundation of the proposed HIDACT algorithm.

Adaptive Computation Time

Adaptive Computation Time (ACT) [151] is first proposed to train RNNs that can adjust the amount of computation for each step. This is achieved by estimating the computational expense at each frame, and the computation would terminate once the accumulated expense exceeds a predefined budget. Formally, at frame t, the neural network computes the expense of processing for the $k-$th step is defined as

$$h_t^{(k)} = f_h(\mathbf{s}_t^{(k)}) \in [0, 1], \tag{3.7}$$

where f_h is a feed-forward layer, and $\mathbf{s}_t^{(k)} \in \mathbb{R}^{D_s}$ is the RNN's step-wise hidden state with a dimensionality of D_s. The computation will halt at $N(t)$ steps once the accumulated expense exceeds the budget:

$$N(t) = \min\{N : \sum_{k=1}^{N} h_t^{(k)} \geq 1 - \epsilon\}, \tag{3.8}$$

where ϵ is a small constant. We define the remainder term $R(t)$ as

$$R(t) = 1 - \sum_{k=1}^{N(t)-1} h_t^{(k)}, \tag{3.9}$$

which stands for the remaining computational budget after processing the input at time t. Alongside the task loss, ACT aims to minimize the following objective

$$\mathcal{L}_{act} = \sum_t N(t) + R(t), \tag{3.10}$$

which encourages the network to produce larger values of $h_t^{(k)}$ at earlier steps k so that budget will be used up within fewer steps, thereby reducing the amount of computation.

We define the aggregate hidden state $\mathbf{s}_t$ and output $\mathbf{y}_t$ at frame t as the weighted sum of the output of individual steps as follows:

$$\mathbf{s}_t = \sum_{k=1}^{N(t)} w_t^{(k)} \mathbf{s}_t^{(k)}, \quad \mathbf{y}_t = \sum_{k=1}^{N(t)} w_t^{(k)} \mathbf{y}_t^{(k)}, \tag{3.11}$$

where the weights $w_t^{(k)}$ follow

$$w_t^{(k)} = \begin{cases} h_t^{(k)}, & \text{if } k < N(t), \\ R(t), & \text{if } k = N(t). \end{cases} \tag{3.12}$$

After all the information is observed at time t, $\mathbf{s}_t$ is passed to the processing of time $t+1$.

Similar to the MVN, after all the information is observed at time t, the aggregate hidden state $\mathbf{s}_t$ is passed to the processing of the next time frame $t+1$, thereby propagating temporal information. It has been empirically shown that ACT allocates more computation on harder-to-predict input and minimizes the computation on easier cases [151]. ACT has been extended to a variety of other tasks, such as early exiting of CNN layers [152, 153] for image classification, adaptive depth Transformers [154] for language modeling, and for multi-channel speech enhancement [155].

Differentiable Adaptive Computation Time

ACT suffers from instability in the number of processing steps and requires extensive hyperparameter tuning for good performance [155, 156]. This issue is attributed to the fact that the ACT pipeline is not fully differentiable. As described in Equations (3.9) and (3.12), ACT imposes the constraint that the intermediate weights $w_t^{(k)}$ should sum to one. If the sum is greater than one, then the weight of the final step is clipped, making the final output non-differentiable.

To overcome this issue, Differentiable Adaptive Computation Time (DACT) [156] is proposed as an end-to-end differentiable pipeline based upon ACT. It applies to halt only at test time while

always processing all information during training. This is achieved by constructing the uncertainty $g_t^{(k)} = f_g(\mathbf{s}_t^k) \in [0, 1]$ at each step k, where f_g is the neural network. The probability that a subsequent processing step $k' > k$ may change the model's prediction is defined to be the product $p_t^{(k)} = \prod_{i=1}^{k} g_t^{(k)} \in [0, 1]$. The loss function is defined as

$$\mathcal{L}_{dact} = \sum_{t}^{T} \sum_{k}^{K} p_t^{(k)}, \tag{3.13}$$

where T is the total number of time frames and K is the maximum number of processing steps. Minimizing Equation (3.13) encourages the model to be confident about its prediction at earlier steps.

The accumulated output $\mathbf{a}_t^{(k)}$ is a linear combination of the step-wise output $\mathbf{y}_t^{(i)}$ from all previous processing steps $i \leq k$:

$$\mathbf{a}_t^k = \begin{cases} \mathbf{0}, & \text{if } k = 0, \\ \mathbf{y}_t^{(k)} p_t^{(k-1)} + \mathbf{a}_t^{(k-1)}(1 - p_t^{(k-1)}), & \text{if } k > 0. \end{cases} \tag{3.14}$$

Notice that if $g_t^{(k)} \to 0$ at step k then $p_t^{(k')} \to 0$ for all subsequent steps $k' > k$; the model's prediction $\mathbf{a}_t^{(k)}$ is hence unlikely to change. This property allows us to define an early stopping condition with a lower bound of $l^{(k)}[c^*]$ for the class c^* with the highest probability and an upper bound $u^{(k)}[c']$ for the runner-up class c' (details in Algorithm 3.1). During inference, we reach the halting condition once $l^{(k)}[c^*] > u^{(k)}[c']$. A theoretical proof of the bounds and the fact that the class label is guaranteed not to change by subsequent processing steps is presented in [156].

Hidden-state Interpolated Differentiable Adaptive Computation Time

DACT is initially proposed for processing non-temporal information and is not concerned with passing hidden states between temporal steps. Hence, it does not perform interpolation in the hidden space. When unrolling across both spectral and temporal dimensions, the hidden state at the last step of each frame feeds into the processing of the next frame. Preliminary experiments show that this formulation leads to significant degradation in test performance. One possible explanation is that the model always processes the maximum number of steps K during training, but the termination condition might be reached earlier during inference with a hidden state distinct from processing all K steps.

We overcome this problem with the proposed Hidden-state Interpolated Differentiable Adaptive Computation Time (HIDACT) by interpolating between the step-wise hidden states, as follows:

$$\mathbf{q}_t^{(k)} = \mathbf{s}_t^{(k)} p_t^{(k-1)} + \mathbf{q}_t^{(k-1)}(1 - p_t^{(k-1)}). \tag{3.15}$$

Algorithm 3.1: Forward pass of HIDACT.

Input: f_s, f_g, f_y, $\mathbf{x} \in \mathbb{R}^{T \times K}$
Output: $\mathbf{a} \in \mathbb{R}^{T \times C}$, $\mathcal{L}$
$\mathcal{L} \leftarrow 0$
for $t \leftarrow 1$ *to* T **do**
  `// Aggregate quantities`
  $\mathbf{q}_t \leftarrow 0$; $\mathbf{a}_t \leftarrow 0$; $p \leftarrow 1$
  for $k \leftarrow 1$ *to* K **do**
    if $k == 1$ **then**
      $\mathbf{s}_t^{(k)} \leftarrow f_s(\mathbf{q}_{t-1}, \mathbf{x}_t^{(k)})$
    else
      $\mathbf{s}_t^{(k)} \leftarrow f_s(\mathbf{s}_t^{(k-1)}, \mathbf{x}_t^{(k)})$
    end
    $\mathbf{y}_t^{(k)} \leftarrow f_y(\mathbf{s}_t^{(k)})$
    $g_t^{(k)} \leftarrow f_g(\mathbf{s}_t^{(k)})$
    `// Update aggregate quantities`
    $\mathbf{q}_t \leftarrow \mathbf{s}_t^{(k)} \cdot p + \mathbf{q}_t \cdot (1 - p)$
    $\mathbf{a}_t \leftarrow \mathbf{y}_t^{(k)} \cdot p + \mathbf{a}_t \cdot (1 - p)$
    $p \leftarrow g_t^{(k)} \cdot p$
    if *training* **then**
      `// Update loss`
      $\mathcal{L} \leftarrow \mathcal{L} + p$
    else
      `// Check halting condition`
      $c_{\text{sort}} \leftarrow \text{argsort}_c \mathbf{a}_t[c]$ `// Descending`
      $c^* \leftarrow c[1]$; $c' \leftarrow c[2]$
      $l[c^*] \leftarrow \mathbf{a}_t[c^*](1 - p)^{K-k}$
      $u[c'] \leftarrow \mathbf{a}_t[c'] + p \cdot (K - k)$
      if $l[c^*] > u[c']$ **then**
        break
    end
  end
end

Similarly, we interpolate the step-wise output vector $\mathbf{a}_t^{(k)}$ with

$$\mathbf{a}_t^{(k)} = \mathbf{y}_t^{(k)} p_t^{(k-1)} + \mathbf{a}_t^{(k-1)} (1 - p_t^{(k-1)}). \tag{3.16}$$

We describe the process of constructing the aggregate hidden state $\mathbf{q}_t$ and output $\mathbf{a}_t$ at each time frame t in Algorithm 3.1. With this interpolation, we impose a constraint on the accumulated hidden state such that it cannot change significantly once the uncertainty $p_t^{(k)}$ becomes small and the halt criterion is met, significantly improving the performance and stability during test time.

3.2.3 Experimental Setup

Dataset

We synthesize a dataset for monophonic sound event detection due to the limited availability of annotated data without overlapping events. Our data is synthesized from the dataset of DCASE 2016 Task 2 [158]. The training split of the DCASE dataset consists of 20 isolated segments for each one of the 11 event classes. We randomly select 16 segments from each class to synthesize our training set and 4 segments for the test set. These segments are downsampled to 16k Hz and mixed with Gaussian noise with the signal-to-noise ratio (SNR) at 0 dB. Each one of the synthesized recordings is 10 seconds long, and the training set contains 1000 recordings while the test set contains 200 recordings. We compute the log mel-spectrogram with 128 features with a window size of 1024 and a hop size of 256.

Model Configuration

Name	# Params (1e6)	# Training bands	# Test bands	Loss type
FB1 FB3	2,633 2,893	8	8	$\mathcal{L}_{ce}$
SB SBR	133	8 random	8	$\mathcal{L}_{ce}$
HIDACT ACT		8 adaptive	adaptive	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{dact}$ $\mathcal{L}_{ce} + \lambda\mathcal{L}_{act}$

Table 3.1: Overview of the proposed HIDACT and baseline systems.

We partition the spectral frames into 8 non-overlapping sub-bands so that each sub-band contains 16 contiguous frequency bins. Similar to the architecture in [9, 10], our sub-band network contains four convolutional blocks with 16, 32, 64, and 128 channels, respectively, followed by a 128-dimensional unidirectional GRU with softmax output heads. We fix the kernel size to be 1×1 to reduce the computation cost, and we perform convolution within each frequency sub-band.

We compare the proposed method with several sub-band and full-band baseline models. The sub-band baseline model (SB) has the same model configuration as the proposed system with the exception that the baseline is trained with only the cross-entropy loss $\mathcal{L}_{ce}$. We also propose a variant of this baseline by masking out a uniformly random number (between 0 and 7) of contiguous frequency sub-bands during training (SBR).

The full-band model (FB1) has the same configuration as the sub-band model except that the GRU contains 2 layers, each with 512 hidden units, to handle the larger input dimension. We also experiment with a version of the full-band model using 3×3 convolutional kernels (FB3). The full-band models are also trained with the cross-entropy loss only.

Table 3.1 shows the comparison between the proposed and each of the baseline models, including the model size, the number of sub-bands processed during training and inference, and the loss type. All sub-band systems have roughly the same number of parameters, and they only differ in the processing of intermediate hidden state and output. The sub-band models have significantly fewer parameters compared to the full-band models. All the baseline models except SBR process a fixed number of frequency sub-bands during both training and test time, while SBR is designed to process a random number of sub-bands during training but also makes a fixed amount of computation at test time. ACT is adaptive to the input during both training and test, while HIDACT always processes all sub-bands during training and adaptively selects the number of sub-bands during inference. All baseline models are trained only with the cross-entropy loss $\mathcal{L}_{ce}$, while the adaptive systems have an additional pondering cost. λ is a hyperparameter controlling the weights of the ponder term and is chosen as 0.001 for ACT and 0.01 for HIDACT in our experiments.

Training and Evaluation Details

During training, we randomly select a 2-second window (128 frames) from the entire input to speed up processing. All models are trained with a batch size of 16 and optimized with Adam [134] using an initial learning rate of 1e-4. We train the networks for a maximum of 100,000 iterations (roughly 1500 epochs), and we apply early stopping if the validation macro-f1 has not improved for 500 epochs.

We evaluate the model's performance on the 200 test recordings using the frame-wise macro F1 score of the 12-way classification, including the background class. The macro F1 score is defined as the mean of the class-wise F1 score. We use the amount of MAC operations [157] as a metric for computational complexity.

3.2.4 Results and Discussions

Comparison with Baselines

We first compare the frame-wise macro F1 score of the test data of the proposed adaptive frequency selection methods (HIDACT and ACT) to the baseline systems in Section 3.2.3. For each model, we make eight different test runs. In each run, we limit the maximum number of frequency sub-bands the model can observe during test time, from 1 to 8, to simulate an environment with a different budget for data and computation. We start with the lowest frequency sub-band and incorporate one contiguous sub-band at a time.

Figure 3.6 (left) shows the curve of F1 versus the maximum number of input sub-bands the model takes as input. Notice that the f1 scores for the baseline models (SB, SBR, FB1, and FB3) always lie on an integer value of the x-axis, since these methods always process all available sub-bands. Among all the sub-band models, the HIDACT model on average processes the fewest frequency

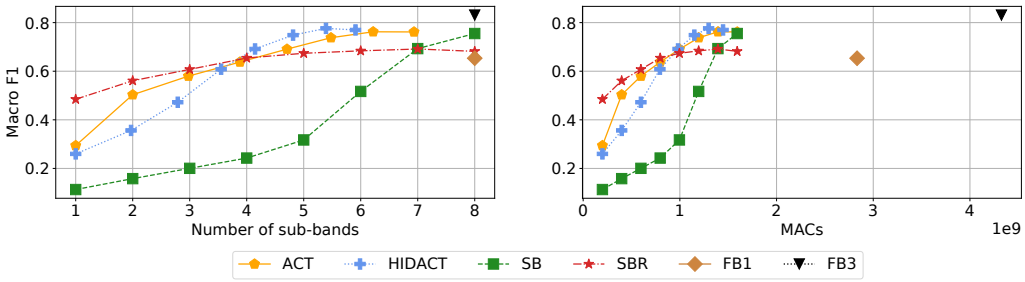

Figure 3.6: Test macro f1 scores for the proposed adaptive computation and baseline systems against the number of frequency sub-bands (left) and the amount of MAC operations (right). The proposed adaptive methods achieve comparable f1 scores within significantly fewer steps and MACs than the baseline models.

sub-bands (less than 6) while having the highest test F1. Both HIDACT and ACT achieve roughly 70% F1 within an average of 5 computational steps per frame. The performance of the baseline sub-band model SB degrades significantly when not observing all 8 sub-bands. SBR, the baseline observing a random number of sub-bands during training, has a less significant performance drop for fewer steps, but the performance with all 8 sub-bands is significantly worse than the other systems. It is possible that SBR overfits an intermediate number of sub-bands and hence the performance stops improving as more input is available.

Figure 3.6 (right) displays the test performance with MACs. Despite the overhead in estimating the uncertainty and the computational cost, these adaptive methods can reach better performance in fewer operations compared to the baseline sub-band SB and the baseline full-band FB1. The performance is comparable to the full-band model FB3 with 3×3 kernels but with much fewer parameters and only one-third of MACs.

Visualization of Adaptive Frequency Selection

We further analyze the behavior of our proposed adaptive computational method HIDACT with respect to the input. Figure 3.7 shows the frame-wise number of frequency sub-bands processed by the HIDACT model, overlayed on the log mel-spectrogram with 128 features with 8 sub-bands of a test recording. Notice the lags of a few frames between the onset of the events and the change in the processing steps. When observing a new event, the network needs to process a few frames to leverage sufficient temporal information. On average, the model requires 6 sub-bands to assign a background label to the frames with no active events; for event classes *coughing, knock*, and *laughter* where the low-frequency components contain strong cues of the event types, the model processes 4-5 frequency sub-bands to make a confident prediction; in contrast, for the *phone* class, most of the rich content and useful information is found at higher frequency bins, and hence the model takes up to 7 processing steps. This diagram indicates that HIDACT not only requires less input and computation but also adaptively adjusts the computation cost based on the input signal.

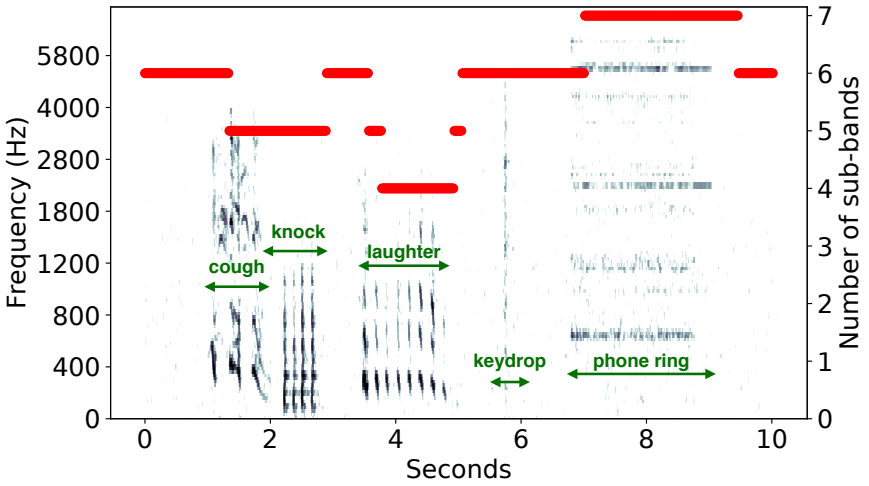

Figure 3.7: The number of sub-bands processed by the HIDACT model for a test recording overlayed on the log mel-spectrogram. We include arrows to indicate the ground-truth event labels. HIDACT can adjust the number of sub-bands it requests and processes based on the spectral properties of the input signal.

3.2.5 Conclusion

We have proposed a method to efficiently detect non-overlapping sound event classes with adaptive frequency selection and sub-band processing. The model is trained once and can adjust its computational cost according to the available computational budget during inference. It adapts to the input signal and learns to make confident predictions without processing all frequency sub-bands. Empirical results show that the proposed HIDACT can maintain the ability to accurately detect sound events when part of the input information is missing, and it achieves comparable performance to full-band or sub-band models with higher computational complexity.

CHAPTER 4: CONTINUAL LEARNING FOR SOUND CLASSIFICATION

In Chapter 3, we perform sound event detection under conventional settings for supervised learning, where the training of neural network models takes place by looping through a stationary dataset until convergence. However, as audio data is being incessantly produced in our daily life, the assumption of training under a fixed dataset may not be applicable in practice. In this chapter, we explore the scenario where we continually update a sound classification system so that it can adapt to the recently collected data samples while retaining its performance on the old data it has previously been trained on. Suppose we want to train a sound classifier for environmental sound classes. We are first given a set of recordings of animal sound classes (e.g., dog barking, rooster crowing), and we train the model for a few epochs until it can effectively discriminate different animal sounds. Then, some sound clips of natural sound (e.g., raindrops, thunderstorms) become available, and we would like to update the model with the recently collected data so that the model can discern both animal and natural sound. One way to do so is to train the model with both the animal and the natural classes, but it is computationally inefficient to train with all the past and current data, and this approach is difficult to be scalable as the size of the dataset keeps growing. An alternative strategy is to train the model with only the new natural sound clips; however, the model may lose the ability to recognize animal sounds as it picks up natural sounds, a phenomenon referred as *catastrophic forgetting* [32] where the performance on previous tasks is impaired as the model gets trained progressively on new tasks.

To balance the tradeoff between efficiency and performance, a plethora of *continual learning* (CL) approaches have been proposed to learn new tasks efficiently while retaining the performance of prior tasks without having full access to the previous datasets. In Section 4.1, we propose to train a sound classifier on a dataset with the number of sound classes increasing over time using the generative replay method, where we continually train a generator alongside the classifier to automatically generate data samples from previous data. Apart from the supervised learning framework, in Section 4.2 we further investigate approaches to incrementally learn representations for sound classes with limited or no annotations to the audio clips.

4.1 CONTINUAL LEARNING OF NEW SOUND CLASSES WITH GENERATIVE REPLAY

4.1.1 Motivation for Continual Learning

Standard supervised machine learning setup posits that the full training dataset is available to the model at once. This is a simplistic assumption. In the wild, the training data may arrive in (non-iid) batches and new classes may appear throughout the learning process. This is typical of human learning where new concepts (classes) are learned throughout life.

[2]Part of this chapter has been published in [39, 53].

Continual learning (CL) proposes a more realistic sequential learning paradigm composed of training episodes [159, 160, 161]. At each episode, the model is only trained on data from a single new task and does not have access to data from earlier tasks. Continual learning is also useful for devices with constrained access to data (either due to storage limitations, or privacy constraints). In such cases classifiers need to be continually trained to learn new classes while minimizing storage. This limits the amount of possible retraining on previous tasks.

Continual learning is particularly challenging for neural networks because of *catastrophic forgetting*: at each episode the network will "forget" the knowledge it has learned in earlier tasks [32]. While several methods have been recently proposed for continual learning [34, 35, 36, 121, 122], much work remains before continual learning becomes a practical technique.

In this section, we explore a continual learning setup for training a classifier on environmental sound classes. This is a challenging task because it necessitates learning a classifier on time-series data, as opposed to typical applications in the continual learning literature that focus on static data (e.g., images) [35]. In addition to being an analog to the way in which human beings learn to classify sounds, the continual learning setup is also useful in practical applications where continual learning of new sound classes needs to be performed by devices with constrained access to data (either due to storage limitations, or privacy regulations).

To alleviate catastrophic forgetting, we utilize the generative replay technique [36], which provides very competitive continually-learned classifiers. A generator is trained simultaneously with the classifier. For each task, the generator is used to simulate earlier-task examples for the classifier. Further, we propose a convolutional autoencoder architecture to embed time-series data, and we make use of the two-step learning framework introduced in [162, 163] to learn the generative model to replay earlier tasks.

We experiment with the ESC-10 (Environmental Sound Classification) dataset [66]. Namely, we compare our proposed generative replay-based method with *rehearsal* which consists in storing a fixed percentage of the data associated with earlier tasks to combat forgetting. Stored data is used as training data in each of the subsequent episodes. This method has been shown to be a very strong baseline [164]. We show that by using a generative model with a size approximately equal to 4% of the whole training set, we are able to match the classification accuracy obtained with a rehearsal method that stores 20% of the training dataset.

4.1.2 Methodology for Supervised Continual Learning

In Section 2.5, we provide a definition of continual learning, a setting in which the goal is to train a model on a sequence of datasets $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T\}$, where each dataset corresponds to a (new) *task* $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T$. According to the standard continual learning setup, when training the model for task t, the data of past tasks and future tasks are not available (we are only allowed to use the dataset $\mathcal{D}_t$). The objective is to learn a single model which is able to predict well on data from all tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$, despite training in a sequential manner. This is challenging in neural network models

as training on the current task without incorporating data from earlier tasks typically results in forgetting the existing knowledge. This phenomenon is referred to as *Catastrophic Forgetting* [32]. Namely, when training for task t, the model forgets the knowledge related to tasks with index $< t$, if no measures are taken to mitigate forgetting. In addition to avoiding catastrophic forgetting, another goal of continual learning is to improve/speed-up learning on future and past tasks. This is referred to as *forward transfer* and *positive backward transfer* [121]. This is a very interesting research direction for continual learning, but in this part of the thesis we focus more on combating catastrophic forgetting. Next, we describe two strategies for alleviating catastrophic forgetting.

Naive Rehearsal

A simple method to combat catastrophic forgetting is to keep a buffer of random samples to help the model remember the past tasks. The buffer contains examples from earlier tasks to reinforce the knowledge from earlier tasks, when training on the current task. This method is referred to as naive rehearsal or simply *rehearsal*, as we do in the rest of this paper. Although simple, this method is surprisingly effective, and has been shown to perform very comparable to state-of-the-art continual learning methods on various standard continual learning experiments [164]. For this reason, we use rehearsal as a baseline method. When training for task $\mathcal{T}_t$, we keep a buffer $\mathcal{M} = \bigcup_{k=1}^{t-1} \mathcal{M}_k$, where $\mathcal{M}_k = \{\mathcal{X}_k^{\text{buffer}}, \mathcal{Y}_k^{\text{buffer}}\} \subset \mathcal{D}_t$ contains randomly selected annotated data from the dataset $\mathcal{D}_k$ associated with task $\mathcal{T}_k$ with $k \leq t-1$. The cost function associated with rehearsal is:

$$\mathcal{L}_{\text{rehearsal}}(\mathcal{D}_t, \mathcal{M}_{1:t-1}, f_t) = \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \mathcal{L}_{\text{clf}}(y, f_t(x)) + \sum_{k=1}^{t-1} \frac{1}{|\mathcal{M}_k|} \sum_{(x',y')\in\mathcal{M}_k} \mathcal{L}_{\text{clf}}(y', f_t(x')), \tag{4.1}$$

where the input features are denoted with x, the target values are denoted with y, f_t stands for the continually trained classifier at task $\mathcal{T}_t$, and $\mathcal{L}_{\text{clf}}$ is the loss for classification, which is typically chosen as the cross-entropy loss. The first term accounts for the loss on the current task (current loss), and the second term accounts for the rehearsal loss. In Figure 4.1 we illustrate the schematics of the loss function.

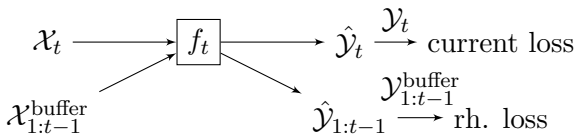

Figure 4.1: The diagram for the loss computation in the naive rehearsal method at task $\mathcal{T}_t$. We separately compute two losses for the current tasks, and a rehearsal (rh.) loss on the stored buffer.

Even though rehearsal combats forgetting, it requires the storage of data in the form of a rehearsal buffer. In the next section, we introduce another method that mitigates forgetting by continually learning a generative model, which does not require storage of past data items.

Generative Replay

An effective alternative to rehearsal is generative replay [36]. This method continually trains a generative model in addition to the classifier to *replay* the data from earlier tasks. By virtue of having a generative model, in lieu of storing examples from earlier tasks, we generate data and use this generated data to avoid forgetting (by using it as training data). The cost function for continual classifier training is therefore written as follows:

$$\mathcal{L}_{\text{genreplay}}(\mathcal{D}_t, \mathcal{D}_g, f_t) = \sum_{(x,y)\in\mathcal{D}_t} \mathcal{L}_{\text{clf}}(f_t(x), y) + \sum_{x_g\in\mathcal{D}_g} \mathcal{L}_{\text{clf}}(f_t(x_g), f_{t-1}(x_g)), \tag{4.2}$$

where the first term is the loss associated with the current task, and the second term is the loss associated with the rehearsal, where $\mathcal{D}_g$ is data simulated from the generative model G_{t-1} after being done with training it until task $t-1$, which is used to rehearse the datasets $\{\mathcal{D}_1, \ldots, \mathcal{D}_{t-1}\}$. The schematic illustration of this loss function is shown in Figure 4.2.

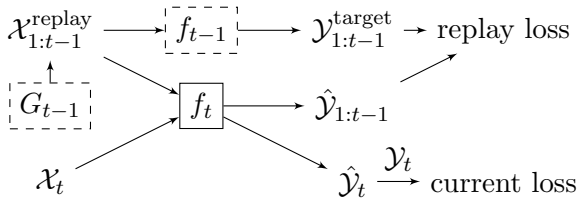

Figure 4.2: Training the classifier using the generative replay at task $\mathcal{T}_t$: The data for the earlier tasks is generated from G_{t-1}. The outputs of the current classifier f_t and the earlier classifier f_{t-1} are matched to compute a replay loss. The weights of the modules in dashed blocks are kept frozen, and only the modules in solid blocks are updated.

Similarly, the generator G_t is trained by using the examples from the current dataset $\mathcal{D}_t$ and the simulated examples from the generator G_{t-1}:

$$\mathcal{L}_{\text{gen}}(\mathcal{D}_t, \mathcal{D}_g, G_t) = \sum_{x\in\mathcal{D}_t} \mathcal{L}_{\text{gen}}(x) + \sum_{x_g\in\mathcal{D}_g} \mathcal{L}_{\text{gen}}(x_g), \tag{4.3}$$

where again the loss function is composed of the current loss term (the first term) and the rehearsal loss (the second term). In both (4.2) and (4.3) we choose $|\mathcal{D}_t|$ and $|\mathcal{D}_g|$ such that the weight of each task is equal. We illustrate the workflow of the method in Figure 4.3.

Note that in our application the generator G produces spectral frames since our goal is to classify segments of audio data. Next, we describe the details of the generative model and architecture for G.

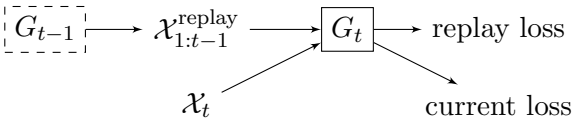

Figure 4.3: Diagram for continually training a generative model using generative replay: At task t, the data is *replayed* from the generative model G_{t-1}, and its likelihood is evaluated on the generative model G_t that we currently train.

Generative Model

In this paper, we use maximum-likelihood-based generative modeling as opposed to Generative Adversarial Networks (GANs) [109] as the former is significantly easier to train [165]. In our generative models, we use a convolutional autoencoder to compute embeddings for spectrogram sequences. The architecture of our autoencoder is shown in Figure 4.4, which consists of using convolutional layers across the time axis to model the temporal structure and then reducing and increasing the feature dimensionality using fully connected layers. After learning the embeddings h, we learn the generative model by fitting a Gaussian mixture model (GMM) on the latent embeddings, as described in the 2-step learning method in [162]. The advantages of using GMMs in the latent space (as opposed to using a standard Gaussian prior like the standard VAE [110]) is advocated by multiple papers in the literature [162, 166, 167, 168, 169]. In our experiments, we observed that separating the learning of parameters of the prior distribution on the latent variables from the learning of the autoencoder resulted in the accurate learning of the generative model (which we refer to as 2-step training). We have observed that the joint training of GMM and the autoencoder often resulted in slightly worse results than that of the 2-step learning approach, and therefore we have chosen to use the 2-step training rather than jointly training the prior and the autoencoder. We also compare the proposed generative modeling scheme with VAEs with standard Gaussian prior [110], and observe that the proposed generative modeling scheme yields many superior generations, which results in better classification.

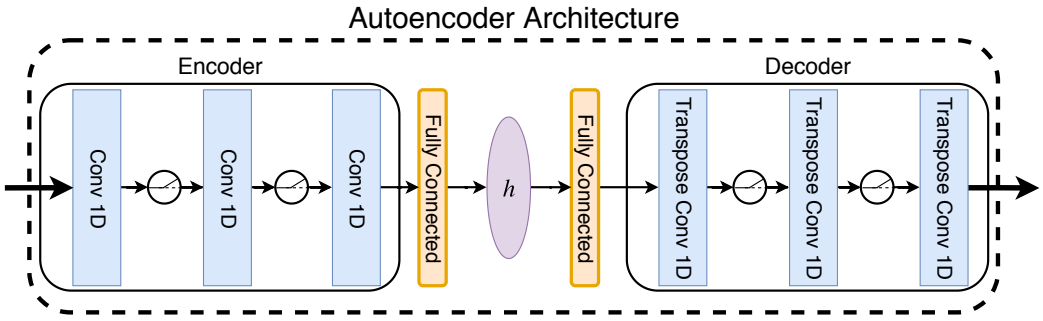

Figure 4.4: The autoencoder architecture used to model the spectra. The convolutional encoder maps the spectra into latent space h, which is then transformed by the decoder into a reconstructed representation. We apply ReLU after each of the first two convolutional layers in both the encoder and the decoder.

4.1.3 Experimental Setup

In this section, we introduce our continual learning setup for audio classification. The experiments simulate scenarios where the model incrementally learns new sound classes without having full access to the previously-encountered sound classes. The model observes ten sound classes in a sequence of five tasks, where in each task two new classes are presented. This is similar to the setup in [170].

Dataset

We select the publicly available ESC-10 [66] dataset for our experiments. The ESC-10 dataset consists of 400 five-seconds recordings sampled at 44kHz of acoustic events from 10 classes, namely: *chainsaw*, *clock ticking*, *crackling fire*, *crying baby*, *dog*, *helicopter*, *rain*, *rooster*, *seawaves*, and *sneezing*.

For each recording, we extract a Time-Frequency spectrogram representation using a 2048 samples window and a 512 samples hop size. Next, we compute the square root of the mel-scaled spectrogram using 128 mel-features for each spectrogram. We further segment our data to snippets that correspond to ≈ 220 ms so that each input data sample has a size of 128×16. We ignore low-energy spectra whose Frobenius norm is less than 1e-4. Finally, we normalize each spectrogram by the maximum energy from each mel-spectrogram so that each value lies in $[0, 1]$. Our initial experimental results demonstrate that normalized mel-spectrograms are more discriminative under the chosen classifier architecture and can be easily reconstructed from the generator. In total there are 9500 mel-spectrograms that we further split into training, validation and test set with a ratio of $7 : 2 : 1$.

To set up the experiment in the setting of continual learning, we partition the dataset into five subsets/tasks where all classes are mutually exclusive. We group the classes based on the their label indices (i.e, *dog* with *rooster*, *rain* with *seawaves*) so the two sound classes from the same group are more similar to each other compared to classes from the other groups. We randomly permute the group order in different runs.

Model Architecture

The classifier contains two 1-D convolutional layers with 64 and 128 filters, respectively, one average pooling layer and two fully-connected layers with 50 and 10 hidden nodes each. For the convolutional layers, we use a filter of length 3 and perform same-padding to the input. We use a rectified linear unit (ReLU) as a nonlinearity after each convolutional layer and the first fully-connected layer. The output of the second fully-connected layer is passed into a softmax layer for a 10-class classification.

For the generator, we experiment with both the autoencoder and the variational autoencoder architectures. The encoder consists of three 1-D convolutional layers followed by a fully-connected

layer with 50 hidden units. Each of the convolutional layers uses 128 filters of lengths 6, 4, and 3 and strides of 1, 2, and 2, respectively. The decoder consists of three 1-D transposed convolutional layers, each with 128 filters of length 4, 4, and 7 and stride of 2, 2, and 1, respectively. We do not perform zero-padding and we apply ReLU after each one of the first two convolutional layers in both the encoder and the decoder as shown in Figure 4.4. For the autoencoder, we train a 50-component GMM for the latent embedding, where each Gaussian component is parameterized by a diagonal covariance matrix. The variational autoencoder architecture contains an additional linear layer on top of the convolutional encoder with 50 dimensions with a reparameterization trick for being able to sample from the latent space.

Training Setup

We compare the proposed generative replay mechanism with rehearsal-based methods. We set up the rehearsal data by storing $p\%$ of the training data at each task into a buffer. This buffer is available to the models throughout all tasks. In our setting, the size of the buffer increases linearly with the number of tasks. We adjust the percentage of the rehearsal data such that the data from each task have an equal probability to be drawn. The parameter of the percentage p of the rehearsal data lies in $p \in \{5, 10, 20, 100\}$.

For all experiments, we optimize our models using the Adam optimizer The batch size is set to 100, and there are 10 - 15 batches per epoch for each task. To train the classifier, we use an initial learning rate equal to 5e-4 and we train it for 300 epochs by minimizing the cross-entropy loss for each task. Moreover, in order to train the generator, we use an initial learning rate of 1e-3 and train it for 1700 epochs for each task. The autoencoder loss is the binary cross-entropy for each time-frequency bin between the original spectrogram and the reconstruction. The loss for the variational autoencoder is the sum of binary cross-entropy and KL divergence between the modeled distribution and unit Gaussian.

4.1.4 Results and Discussions

We report the performance of various replay strategies under the sound classification setup. For each experiment, we report the performance obtained by the models using five different permutations of the order of tasks. In each task, we report the mean accuracy on the test set, which contains all sound classes that the model has seen up until the current task.

Overall Results

Figure 4.5 shows the test accuracy of different generative replay strategies and rehearsal methods for various buffer sizes. "AE+GMM" refers to the proposed generative replay setting with an autoencoder and a Gaussian mixture learned in two steps as described in Section 4.1.2. "VAE"

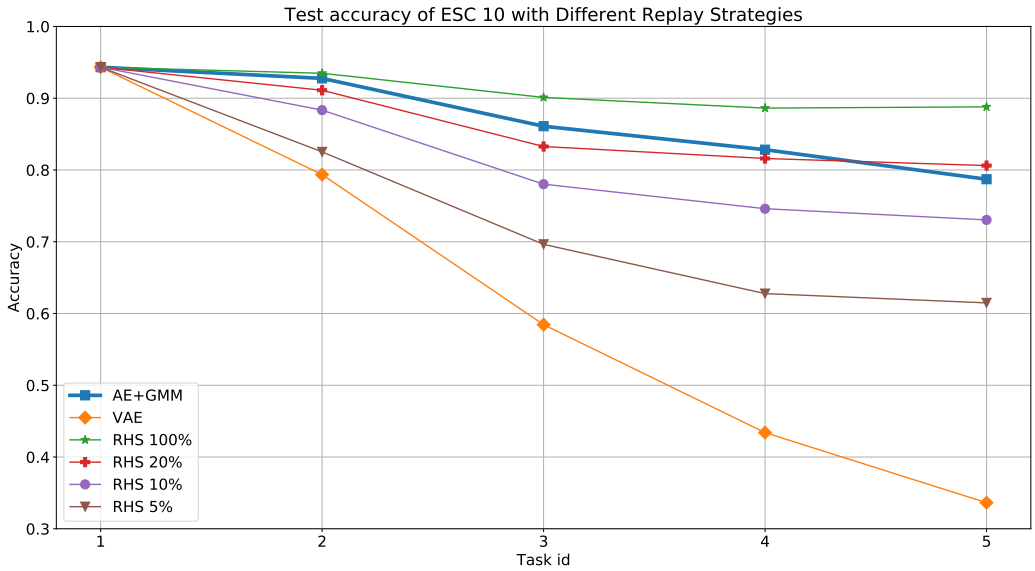

Figure 4.5: Test accuracy on the ESC-10 dataset using generative replay and rehearsal methods with various buffer sizes. The x-axis and y-axis denote the task index and the model's test accuracy, respectively. Each point represents the mean of the accuracy after five runs using different permutations for the tasks.

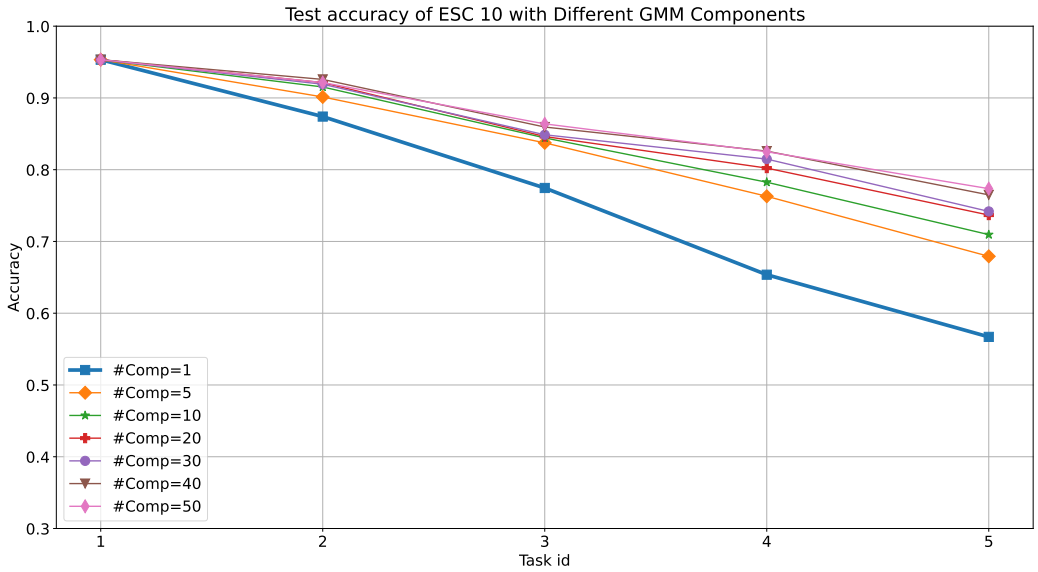

Figure 4.6: Test accuracy on the ESC-10 dataset using AE+GMM over different numbers of Gaussian components.

corresponds to the variational autoencoder mentioned in Section 4.1.3. "RHS X%" denotes a rehearsal-based method with X% training data stored in the buffer. "RHS 100%" is used as an upper-bound estimation of the performance of any replay strategy since it corresponds to the ideal case where all the training data is available in all future stages.

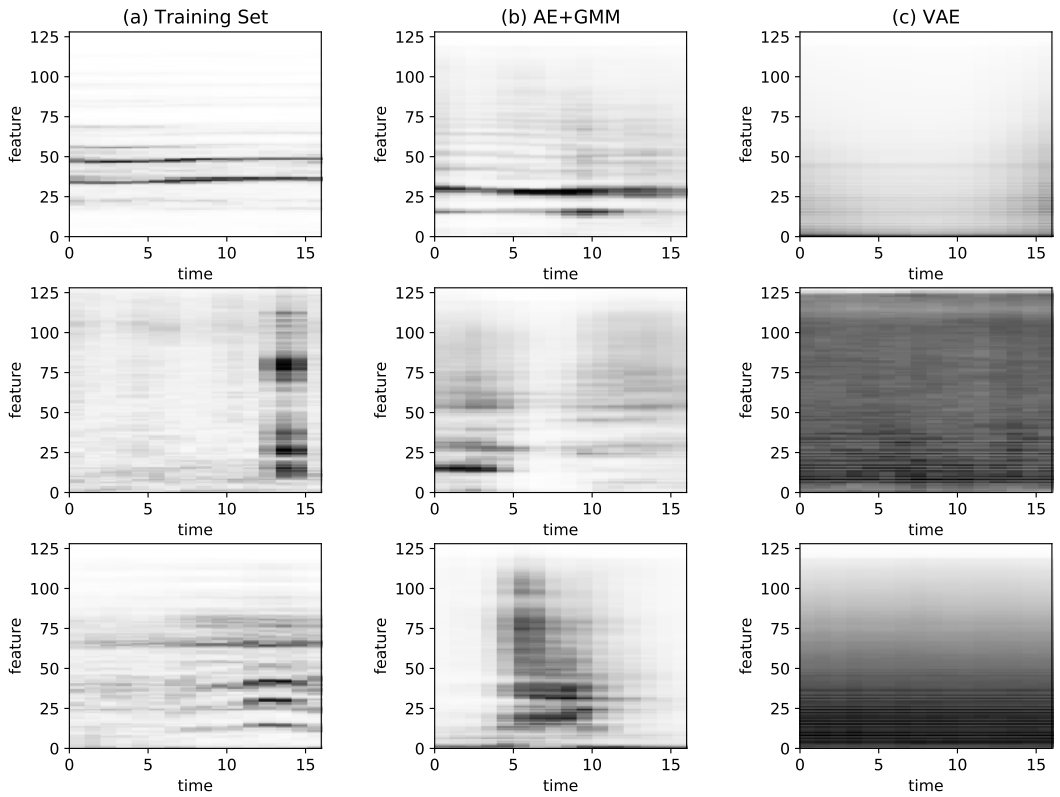

Figure 4.7: Mel-spectrograms from the training set (column a), generated by AE+GMM (column b) and by a VAE (column c). Notice how the VAE generated data do not reproduce salient class features, whereas the proposed AE+GMM generator does so better.

Overall, RHS 100% has the highest mean accuracy and it fits the expectation as an upper-bound estimation of any replay strategy. The performance of rehearsal methods increases as the proportion of the data stored in the buffer increases. We also notice that for all methods the variance of the test accuracy tends to decrease as the number of tasks increases. Initially, the variance is large because a random binary classification task might deviate too much in terms of difficulty from another. However, towards the end, the models have seen all sound classes regardless of the permutation, and therefore the mean accuracy tends to stabilize.

Impact of Two-step Learning and Number of Gaussian Components

We next assess the choice of the generative model between the conventional VAE and the proposed two-step AE+GMM. From Figure 4.5, we can observe that AE+GMM significantly outperforms VAE as a replay strategy throughout the sequence of tasks. The average accuracy after the final task for AE+GMM (78.7%) is similar to RHS 20% (80.6%), while VAE (33.6%) performs significantly worse than 5% (61.5%). We analyze such notable differences by looking at the samples generated by both models as illustrated in Figure 4.7. We show three examples from the training set and the respective generations using AE+GMM and VAE. Note that VAE smooths out the temporal

structure of the generated mel-spectrograms and lacks diversity between classes. On the other hand, AE+GMM generates mel-spectrograms with a much more diverse temporal structure, exhibiting a much closer resemblance to the examples from the training set.

To further examine the factors leading to AE+GMM's superior performance over conventional VAE, we perform an additional ablation study on the number of Gaussian components for the two-step training. To highlight the impact of two-step learning, we include the results when the latent space of the two-step model is parameterized with only one Gaussian with diagonal covariance, which is equivalent to the latent space parameterization of the conventional VAE, as shown in the curve in Figure 4.6. Under the single Gaussian, the two-step model obtains an accuracy of 56.7% at the end of the final task, surpassing the jointly trained VAE by 23.1%. The accuracy values increase as the number of Gaussian components rises, but the performance gain diminishes beyond 20 components. We did not experiment with over 50 components since the training of the GMM becomes unstable and less likely to converge.

Comparison Between AE+GMM and Rehearsal Based Methods

We observe that AE+GMM performs significantly better than rehearsal schemes with buffer proportion $p = 5\%, 10\%$. The accuracy of AE+GMM is almost identical to RHS 20% at the last task and marginally higher in all previous tasks. The total number of trainable parameters in AE+GMM is less than $480,000$. The size of the network is equivalent to $\frac{480000/(128\times16)}{9500\times0.7} \approx 3.5\%$ of the training data. In other words, using a generator whose size is less than 4% of the training data, we are capable of reaching the accuracy comparable to storing 20% of the data. The result demonstrates the effectiveness of AE+GMM generative replay strategy when limited storage space is available.

4.1.5 Conclusion

We showed that generative replay is an effective continual learning method for audio classification tasks. Using a generative model whose size is less than 4% of the size of the training data, we obtain a test accuracy comparable to a buffer-based rehearsal scheme which needs to store 20% of all used training data. These results highlight the potential of using generative models instead of keeping previously seen training data when there are storage constraints. We see these aspects as being crucial to sound recognition systems for which keeping prior training data is prohibitive, but often need (to learn) to perform new tasks on the fly.

4.2 LEARNING REPRESENTATIONS FOR NEW SOUND CLASSES WITH CONTINUAL SELF-SUPERVISED LEARNING

4.2.1 Motivation for Continual Representation Learning

Deep neural networks for audio classification require large amounts of data to be trained on [17, 50]. However, in many realistic deployments, additional labeled data for new sound classes might present itself after initial training, necessitating a time and resource-consuming retraining process that incorporates both past and new data. This problem could be exacerbated by storage constraints, hardware corruption, or privacy regulations that can limit access to past data. As introduced in Section 2.5, continual learning has been a field of rising interest to address the aforementioned concerns. The goal is to design learning algorithms that would continuously learn from a sequence of tasks to let the models imitate the learning process of human beings. A major problem that arises when a model is continually trained is called *catastrophic forgetting* [32], which is associated with deteriorating performance on previously learned tasks.

In Section 4.1, we present an approach based on generative replay to combat catastrophic forgetting when learning sound classes under a supervised continual learning setup. In this section, we extend this setup to unsupervised learning. We propose to adopt continual representation learning (CRL) for the sound event classification task and experimentally show its efficacy for class-incremental continual learning. Representation learning decouples the learning into two stages: i) Learning of an encoder with a representation-specific objective (e.g., with a self-supervised learning objective). ii) Finetuning of a shallow classifier on top of the encoder for a downstream task. To the best of our knowledge, this is the first time that continual representation learning has been used in the sound event classification domain. We argue that using a representation learning pipeline in continual learning is especially beneficial for practical use-case implications. Namely, i) The majority of the computational burden is passed on to the encoder learning stage, and therefore the shallow output layer can be trained including earlier tasks. ii) Decoupling the output head gives the additional flexibility to add an indefinite number of classes by re-training an ad-hoc output head for each task, which is required in real-life applications. iii) In the case where labeled data is scarce, learning an encoder from unlabeled data is beneficial for generalization performance, as we showcase with our experiments.

As hinted above, we adopt self-supervised learning (SSL) within CRL. As mentioned in Section 2.3, SSL is a relatively novel research area that is revolutionizing deep learning due to its impressive ability to learn features that generalize well without labels. In SSL, deep networks are trained with pretext tasks such as clustering [171], mutual information maximization [172], image colorization [40], masked token prediction [42, 44], contrastive learning [45, 46, 47] and so on. These early works on self-supervised learning focused on images and natural language processing. Only recently has self-supervised learning been extended to audio and speech as well [48, 49, 50, 51, 173, 174].

In the space of incorporating representation learning within a continual learning setup, recent

works include [175, 176, 177, 178]. Different from our work, these works are applied to images and involve explicit ways to combat catastrophic forgetting. A continual representation learning method for speech recognition is proposed in [179] by learning more languages over time, but the size of the model grows as more languages are involved, and language information is required during inference. In contrast, we focus on a more challenging setup where the model remains constant in size while progressively learning new sound classes, and no prior information on the input data is needed at test time (e.g., category of the sound).

To summarize, in this section, we propose using the continual representation learning paradigm for class-incremental learning of new audio classes. We list our contributions as follows:

- We show that adopting representation learning for continual learning of new sound classes enables the study of the practically interesting case where only a small amount of labels is present. This is a realistic use case where the system is mainly trained on unlabeled data. To the best of our knowledge, this is the first study to explore continual representation learning in the audio domain with a partially labeled dataset setup. We investigate both in-domain and out-of-domain performance. For the evaluation of the partially labeled use case, we propose several evaluation metrics.

- We empirically observe that even if we do not employ an explicit mechanism to combat forgetting, employing similarity-based self-supervised learning within CRL yields better performance than continual supervised representation learning, and comparable performance to distillation-based continual learning methods to combat forgetting, which requires additional storage and computation.

4.2.2 Methods for Continual Representation Learning

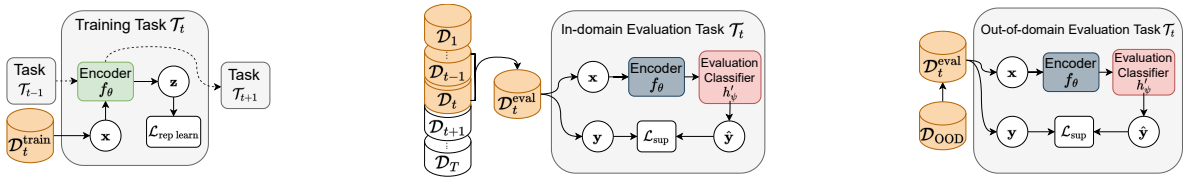

Figure 4.8: **(Left) Training pipeline for continual representation learning:** Representations are learned by continually training an encoder f_θ with one task/dataset at a time. **(Middle) In-domain evaluation:** The representations are evaluated by training a classifier h'_ψ using a labeled training set on top of the frozen f_θ. **(Right) Out-of-domain evaluation:** The dataset used for training the output head h'_ψ is different from the dataset for encoder training.

State-of-the-art representation learning approaches require a large dataset to train high-quality representations. In practical cases where the dataset grows over time, frequently updating the model using all of the data samples is computationally prohibitive. The proposed continual representation learning framework circumvents this bottleneck by training an encoder only for the most recent

task, which carries the majority of the computational burden. For instance, suppose we collect new data with new urban sound types (e.g., car horn, gunshot) to train a model that has been previously trained to recognize animal sounds (e.g., dog, rooster). Instead of retraining the network on both datasets, the encoder is updated only on the urban sound classes while preserving the ability to effectively represent animal classes. Once the encoder is updated with new **unlabeled** data, a shallow output layer is trained from scratch on top of the encoder, with available **labeled** data.

That is, in continual representation learning, we aim to learn low-dimensional embeddings by training an encoder model f_θ on one task at a time. Under the class-incremental setting, in each task, the encoder is trained with data from a set of classes that corresponds to the current task: When entering a new task, f_θ is initialized from the end of the previous task. It is then updated with a representation learning objective with the current dataset $\mathcal{D}_t^{\text{train}}$. After training, the learned representations are evaluated on a classification task. A shallow classifier h'_ψ is trained on top of the encoder's output using a small amount of labeled data $\mathcal{D}_t^{\text{eval}}$ with the encoder's weights fixed. In the case of in-domain evaluation, $\mathcal{D}_t^{\text{eval}}$ contains examples from previous classes (see Figure 4.8), and in the case of out-of-distribution evaluation, $\mathcal{D}_t^{\text{eval}}$ may come from a different distribution from the dataset used for training the encoder. This continual learning framework is shown in Fig. 4.8.

We propose using similarity-based self-supervised algorithms for representation learning in this framework, and we call this approach continual self-supervised representation learning (**CSSL**). As introduced in Section 2.3, these similarity-based SSL methods maximize the similarity between embeddings from a pair of distorted views of the same input to make representations robust to distortions. We consider SimCLR [46, 180], MoCo [45], and Barlow Twins [113] as candidates for the continual representation learning framework. To create positive pairs for these similarity-based methods, we apply audio data augmentation by first taking fixed-length random segments from each clip, and then applying SpecAugment [181] to the spectrograms. We compare CSSL with continual supervised representation learning (**CSUP**), which relies on the assumption that annotations are accessible. In CSUP, we train an additional classifier h_ψ jointly with f_θ to propagate label information. This classifier is discarded for downstream tasks or evaluation. For a fair comparison, we apply identical data augmentations and encoder architecture to both approaches.

We hypothesize that similarity-based SSL is beneficial in the context of CRL due to its strong generalization ability. The set of features learned from the current data generalizes well to past data, and consequently, this results in a system that is less prone to forgetting.

4.2.3 Evaluation for Continual Representation Learning

SSL algorithms are usually trained and evaluated in two steps [45, 46, 113]. First, the encoder network that maps the input to the latent embedding is optimized with the representation learning objectives. Then, the learned embedding is used as input for downstream classification tasks. Similarly, in the continual learning setting, we first learn the representation with the encoder f_θ only using data from the current task. At the end of each task, we evaluate the representation by

training an evaluation classifier, h'_ψ, randomly initialized, on top of the pre-trained frozen encoder (see Fig. 4.8). Note that h'_ψ is much smaller in size compared to the encoder and is computationally more affordable to train. Also, notice that the classifier h_ψ in CSUP (trained jointly with f_θ) is discarded before evaluation since only the classes for the current task are seen during training while all previously seen classes are used for evaluation. Initializing a new evaluation classifier h'_ψ from scratch ensures a fair comparison between CSSL and CSUP.

Following supervised continual learning metrics for classification, for each task $t \in \{1, 2, \ldots, T\}$, we compute the accuracy of the classifier on the test set of task j using the encoder at task t denoted as $A_{t,j}$. We measure the average accuracy at the end of each task: $\bar{A}_t = \frac{1}{t}\sum_i A_{t,i}$ and obtain the average accuracy at the end of the training, $\bar{A} = \bar{A}_T$. Additionally, we compute forgetting, which measures the average decrease in accuracy of each task between its peak and the final performance, defined by $\bar{F} = \frac{1}{T-1}\sum_{j=1}^{T-1} \max_{\tau=1,2,\ldots T}(A_{\tau,j} - A_{T,j})$.

We propose a set of evaluation protocols with different data distributions and types of models for the downstream classification task. When encoder training and evaluation take place on the same dataset (see Fig. 4.8 Middle), we propose the following in-domain protocols:

1. Linear Evaluation Protocol (**LEP**) [177], where a linear classifier h'_ψ is trained with the output of the fixed, pre-trained encoder f_θ using labeled data from past and current task t, $\{\mathcal{D}_\tau\}_{\tau=1}^t$;

2. Subset Linear Evaluation Protocol (**SLEP**), where h'_ψ is trained with a random subset, $\{\mathcal{D}'_\tau\}_{\tau=1}^t$, where $\mathcal{D}'_\tau \subset \mathcal{D}_\tau$. This protocol simulates the scenario with limited labels. We keep a subset of 200 samples per task ($\approx 12\%$ of data).

For both LEP and SLEP, classification becomes more challenging over time since the test data involves more classes as more tasks are learned.

Additionally, we propose an out-of-domain (OOD) protocol (Fig. 4.8 Right) where the representations are evaluated on a different dataset from which they are learned. We introduce the Full Linear Evaluation Protocol (**FLEP**), where at each task the linear evaluation classifier is trained on the full downstream dataset. As opposed to the in-domain protocols, here the test set does not grow with more tasks.

4.2.4 Experimental Setup

Datasets

We use UrbanSound8K [182], the TAU Urban Acoustic Scenes 2019 (DCASE TAU19) [183], and VGGSound [184] to continually train the encoder. We partition each dataset into class-disjoint subsets. Following [164], we set the number of tasks to $T = 5$ or $T = 10$, and the classes are evenly split between tasks. Namely, each task corresponds to a set of classes. For example, in a $T = 5$ case for a 10-class dataset, we have two distinct classes in each task. The ordering of classes is randomly

shuffled before the split. The UrbanSound8K and the TAU Urban Acoustic Scenes 2019 (DCASE TAU19) are used for in-domain evaluation, and VGGSound [184] is used to learn representations for OOD experiments. The details for each dataset are as follows:

- The UrbanSound8K dataset [182] is a dataset for sound event recognition that contains 8,732 recordings with a total duration of 8.75 hours, where the length of each clip is limited to 4 seconds. Each recording is labeled with a single event class from the 10 possible classes, *air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren,* and *street music*. Each class has no more than 1,000 clips. All files are pre-sorted into 10 folds for cross-validation.

- The TAU Urban Acoustic Scenes 2019 dataset is used for the DCASE 2019 Task-1(A) challenge [183] for acoustic scene classification. The development set consists of 40 hours of audio collected from ten cities, with 9,185 recordings for training and 4,185 recordings for evaluation. Each clip is 10 seconds long and is labeled with one of 10 following classes: *airport, shopping mall, metro station, pedestrian street, public square, street traffic, tram, bus, metro,* and *part*. Furthermore, we create the validation set by randomly holding out 20% of the training data for each class.

- The VGGSound dataset [184] is an audio-visual dataset with 200,000 clips with a total duration of 560 hours from more than 300 sound classes such as instruments, horns, and city sounds. The recordings are scraped from YouTube and are labeled by pre-trained image and sound classifiers. However, these data samples may not be accurately labeled due to i) the classifiers are pre-trained on a different dataset, and ii) the recordings may contain interfering sound events while each clip is assigned a single label. We consider learning the representation using VGGSound while evaluating with a different downstream dataset with an OOD setup.

To preprocess the data, we first downsample all audio clips to 16k Hz, and we convert all binaural recordings to a single channel by taking the average of two channels. For each signal, we compute the mel-spectrogram with 80 features with a window size of 25 ms and a hop size of 10 ms. The spectrograms are normalized to be zero-mean and unit-variance across each batch before feeding into the encoder network.

Model Training

We use the CNN14 architecture as the backbone encoder model with 78M trainable parameters [17]. The output representation has 2048 dimensions. We train the encoder for 50 epochs per task on UrbanSound8K and VGGSound, and 100 epochs per task on DCASE TAU19. The linear classifiers for evaluation are trained for 30 epochs per task on UrbanSound8K and 100 epochs on DCASE TAU19. Results are averaged across 10 folds on UrbanSound8K and across three runs

with different seeds on DCASE TAU19 and VGGSound. Additional details can be found in our SpeechBrain [185] implementation[3].

4.2.5 Results and Discussions

	CSUP	SimCLR	Barlow Twins	MoCo
UrbanSound8K	**80.9**	74.3	73.3	69.3
DCASE TAU19	**68.2**	62.5	58.8	50.3

Table 4.1: Offline accuracy when representations are trained with the entire dataset. The best performances are highlighted in bold.

Method	UrbanSound8K ($T=5$)				DCASE TAU19 ($T=5$)			
	LEP		SLEP		LEP		SLEP	
	A (↑)	F (↓)	A (↑)	F (↓)	A (↑)	F (↓)	A (↑)	F (↓)
No distillation								
CSUP	65.6	19.6	48.4	33.1	48.2	27.6	32.5	38.4
SimCLR	**70.3**	15.3	**50.3**	26.6	**59.7**	**17.8**	**42.1**	27.9
Barlow Twins	68.5	**14.1**	49.7	**20.5**	55.9	19.0	41.0	**23.4**
MoCo	68.4	15.6	**50.3**	25.8	49.5	20.2	34.8	25.8
With distillation								
CSUP + $\mathcal{L}_{\text{MSE}}$	58.6	27.0	43.8	39.2	49.1	26.1	35.1	35.8
CSUP + $\mathcal{L}_{\text{sim}}$	70.6	**13.8**	54.9	27.1	56.2	19.7	42.1	32.4
CSUP + $\mathcal{L}_{\text{KLD}}$	69.8	15.9	**55.4**	27.4	55.7	19.4	42.6	30.3
SimCLR + $\mathcal{L}_{\text{MSE}}$	**70.9**	14.6	50.6	25.1	56.2	19.6	42.0	26.5
SimCLR + $\mathcal{L}_{\text{sim}}$	70.6	14.0	51.1	**25.0**	**60.0**	**17.6**	**42.8**	**25.9**

Table 4.2: In-domain evaluation: Average accuracy (A) and forgetting (F) for CSSL and CSUP methods. Best performances are in bold. Note that rows with gray backgrounds correspond to the CSSL approaches where no distillation against forgetting is used.

Offline Evaluation

We first train and evaluate the representations with supervised and SSL algorithms under the ideal offline scenario in which the entire training set is available for training (i.e., there is only $T = 1$ task). We consider three SSL algorithms, namely, SimCLR, MoCo, and Barlow Twins. For evaluation, we train a new classifier h'_ψ on the output of the fixed, pre-trained encoder f_θ following Section 4.2.3.

[3] https://github.com/zhepeiw/cssl_sound

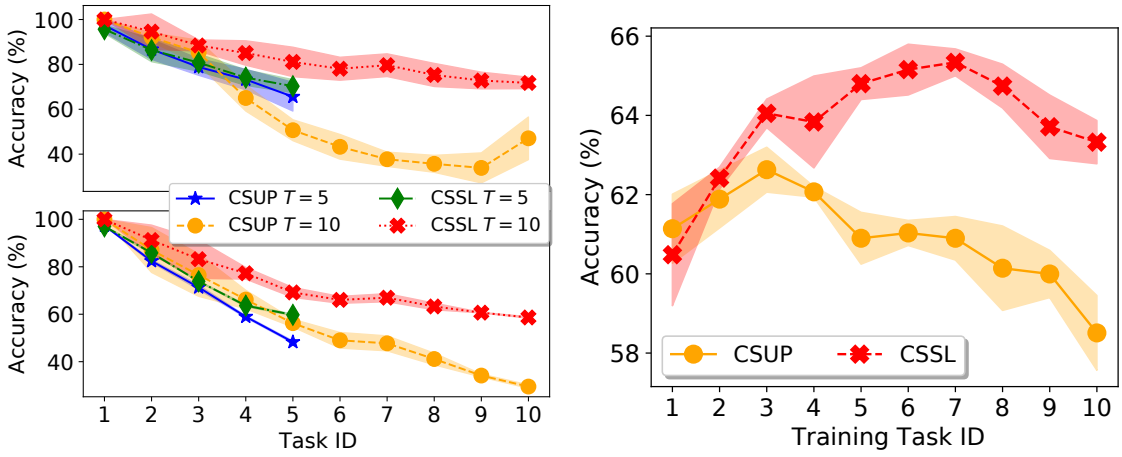

Figure 4.9: **(Left) The In-Domain experiment:** The classification performance obtained with CSSL and CSUP on UrbanSound8K (top) and DCASE TAU19 (bottom). **(Right) The OOD experiment:** Performance of CSSL and CSUP trained on VGGSound and evaluated with full-set linear evaluation protocol (FLEP) on DCASE TAU19.

As we show in Table 4.1 on both datasets, CSUP outperforms the CSSL approaches. This indicates that the supervised classification loss provides a strong baseline for representation learning when trained with the entire corpus at once without continual learning constraints.

Comparison between CSSL and CSUP

We compare representations learned from CSSL and CSUP in terms of their classification performance and resilience to forgetting. Similar to the offline experiments, we consider SimCLR, MoCo, and Barlow Twins for CSSL. In this experiment, we perform an in-domain evaluation with $T = 5$ tasks, and we measure both values for average accuracy and forgetting at the completion of all tasks.

Table 4.2 (No distillation) summarizes the average accuracy and forgetting for the in-domain protocols. First, the accuracy values for all learning methods for $T = 5$ are lower than the corresponding offline accuracy in Section 4.2.5; this indicates the existence of catastrophic forgetting for continual representation learning regardless of the approaches. While supervised learning beats SSL in offline evaluation, the trend is reversed when evaluating in a continual learning setup. All three CSSL methods (highlighted in gray) outperform CSUP on both datasets across all evaluation protocols and metrics. The consistent advantage of SSL in both accuracy and forgetting indicates that **the similarity-based objectives learn features that generalize better and are less affected by forgetting than CSUP**.

In Fig. 4.9 (Left), we also show the trajectories of accuracies obtained with CSSL and CSUP for five tasks ($T = 5$) and ten tasks ($T = 10$) using SimCLR as the CSSL method. CSSL has higher average accuracy, and generally, the gap widens for both $T = 5$ and $T = 10$ as training progresses.

Knowledge Distillation

We additionally study the effects of regularization-based CL algorithms by adapting the Learning Without Forgetting (LwF) [34] framework to the continual representation learning setting. At each task $t > 1$, the representation learning objective is optimized jointly with a knowledge distillation loss

$$\mathcal{L}_{\text{KD}}(\mathbf{x}, t) = \mathcal{L}_{\text{dist}}(f_\theta(\mathbf{x}), f_{\theta^{(t-1)}}(\mathbf{x})), \tag{4.4}$$

where $f_{\theta^{(t-1)}}$ is the snapshot of the encoder at the completion of task $t-1$ whose weights are frozen at the current task t. We consider the following candidates for the distillation loss $\mathcal{L}_{\text{dist}}$:

- mean squared error, $\mathcal{L}_{\text{MSE}}$ [186];
- similarity-based objective that is used in SimCLR [112], $\mathcal{L}_{\text{sim}}$;
- KL-Divergence loss, $\mathcal{L}_{\text{KLD}}$ (used for CSUP only) applied to the logits obtained after the output head [34].

Table 4.2 (With distillation) shows the in-domain evaluation results of CSSL with SimCLR and CSUP with these loss options. To verify the consistency of these results for different SSL algorithms, we also tried Barlow Twins and MoCo and observed similar performance. We omit these results due to space constraints.

We find that distillation with $\mathcal{L}_{\text{MSE}}$ does not improve the results for CSUP on UrbanSound8K nor SimCLR on DCASE TAU19. $\mathcal{L}_{\text{KLD}}$ outperforms $\mathcal{L}_{\text{MSE}}$, but it is still beaten by the plain SimCLR except for the SLEP evaluation protocol, even though the plain SimCLR does not require storing any models or labels. Both $\mathcal{L}_{\text{MSE}}$ and $\mathcal{L}_{\text{KLD}}$ explicitly restrict the output to mitigate forgetting, but they reduce the plasticity of learning knowledge from new tasks and hence do not improve the overall performance. With the similarity-based loss $\mathcal{L}_{\text{sim}}$, the accuracy of CSUP increases significantly on both datasets. The performance gain of CSUP once again highlights **the generalization ability of the similarity-based self-supervised objective and its resilience against forgetting in continual representation learning**. The marginal improvement with $\mathcal{L}_{\text{sim}}$ on SimCLR shows that the similarity-based framework alone learns features that generalize across tasks as well as distillation-based methods. For CSSL, distillation does not provide more advantages of generalization despite the cost of additional computation during training and storage for model saving. **Given the computational benefits, we hence conclude that CSSL without explicit methods for combating forgetting is preferable over the alternatives that use distillation.**

Out-of-domain (OOD) Evaluation

In addition to the in-domain evaluation, we compare CSSL and CSUP when the representation learning and downstream evaluation are performed on different datasets. Fig. 4.9 (Right) shows

the trajectories of the average accuracy using the FLEP protocol on DCASE TAU19 when the encoder is trained on VGGSound. We consider SimCLR for CSSL using $T = 10$ tasks. When evaluating the representation using a linear classifier with the FLEP protocol, we see that CSSL outperforms CSUP after the second task. The performance gap widens as the encoder learns more tasks. We observe a decreasing trend in the accuracy curve of CSUP, indicating that the transfer of knowledge is overwhelmed by catastrophic forgetting. For CSSL, the curve keeps increasing for the first seven tasks. The results from the OOD evaluation demonstrate the effectiveness of CSSL to learn continually without labels and its ability to generalize to a different downstream task.

Continual Representation Learning with Full Replay Buffer

Table 4.3: In-domain Evaluation for Different Encoder Training Dataset

	CSUP-O	SimCLR-O	CSUP-NR	SimCLR-NR	CSUP-FR	SimCLR-FR
US8K	80.9	74.3	65.6	70.3	**82.3**	77.2
DCASE	68.2	62.5	48.2	59.7	**69.1**	67.4

Table 4.4: Final average accuracy with the linear evaluation protocol when representations are trained with the full dataset at once (CSUP-O, SimCLR-O), continually trained without using data buffer (CSUP-NR, SimCLR-NR), and using full data buffer (CSUP-FR, SimCLR-FR) for $T = 5$ tasks.

We also consider continually training the encoder f_θ and the output head h_ψ with a full replay (FR) buffer by storing data samples from the current and all previous tasks with $T = 5$ tasks. We provide the final average accuracy using the in-domain linear evaluation protocol for UrbanSound8K and DCASE TAU19 in Table 4.4 for CSUP and CSSL using SimCLR, denoted by CSUP-FR and SimCLR-FR, respectively. We also include the corresponding offline systems (denoted by CSUP-O, SimCLR-O) from Table 4.1 and the continually trained systems without using replay buffer (trained only on the current task) - (denoted by CSUP-NR, SimCLR-NR) taken from Table 4.2. Both FR systems achieve the best performance compared to the offline and NR systems by beating the offline systems by a small margin (possibly due to a curriculum effect) and significantly exceeding the accuracy of the NR systems. When the encoder observes data from all tasks in both offline and full replay scenarios, the supervised algorithm slightly outperforms SimCLR. However, this performance gain comes at the extra computational cost of storing and revisiting past data samples during the encoder training. Also, notice that the gap between CSUP-FR and CSUP-NR is significantly larger than the one between SimCLR-FR and SimCLR-NR, and this further indicates that representations learned with supervised objectives are more prone to performance degradation when access to previous data is restricted.

Assessing the Impact of Self-Supervised Objectives

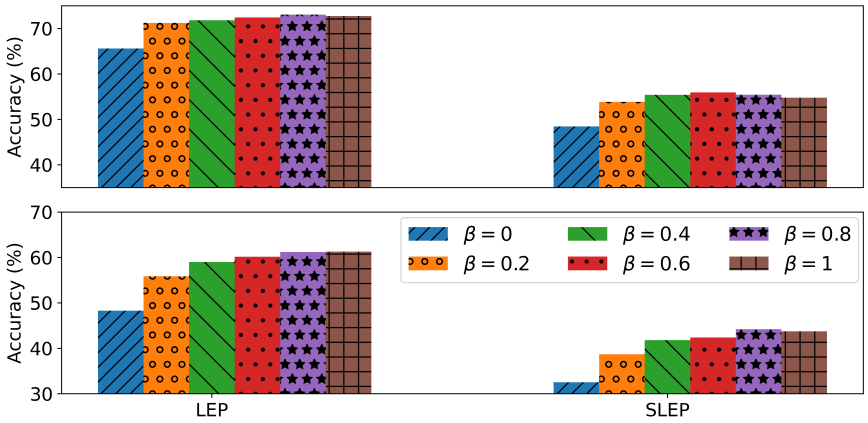

Figure 4.10: Average accuracy on the linear combination of CSSL and CSUP with the supervised weight $\alpha = 1$ fixed on in-domain linear evaluation (LEP) and subset evaluation (SLEP) protocols on UrbanSound8K (upper) and DCASE TAU19 (lower).

To analyze the influence of the SSL objective on continual representation learning, we first perform an ablation study by controlling the amount of self-supervision in the learning framework. For this experiment, we assume the existence of annotations and train the encoder jointly with both classification loss and the similarity-based SSL loss [112], given by

$$\mathcal{L}_{\text{joint}} = \alpha \mathcal{L}_{\text{sup}} + \beta \mathcal{L}_{\text{ssl}}, \tag{4.5}$$

where $\alpha, \beta \geq 0$ control the weight of each objective. We fix the weight of the supervised loss as $\alpha = 1$ and increase the weight of the SSL objective β from 0 to 1. Notice that $\alpha = 1, \beta = 0$ is equivalent to CSUP. We compute the final average accuracy at the end of $T = 5$ tasks using the in-domain protocols on two different datasets, as shown in Figure 4.10. In general, higher weights on SSL leads to better performance. In particular, there is a significant performance gain between $\beta = 0$ and $\beta = 0.2$ across both protocols. These observations imply the benefits of incorporating similarity-based SSL objectives in continual representation learning even if label information is available.

Impact from the Number of Tasks

We are interested in whether the relative performance between CSSL and CSUP is consistent with a varying total number of tasks. As mentioned in Section 4.2.5, Fig. 4.9 (Right) displays the trajectories of the average accuracy at the end of each task for CSSL using SimCLR and CSUP with $T = 5, 10$ total tasks. CSSL has a higher average accuracy at the completion for both $T = 5, 10$ on both datasets than CSUP. The performance gap is significantly larger in $T = 10$ than in $T = 5$. Furthermore, in $T = 10$, this gap widens as training progresses. Since both

UrbanSound8K and DCASE TAU19 contain 10 classes, $T = 10$ creates an extreme setting where the encoder learns only one class at a time. If training without replay data using cross-entropy, the supervised framework may optimize by biasing the weight towards the current class without learning useful representations. In contrast, without label information, the similarity-based CSSL framework is more robust to the shift in data distribution across tasks.

4.2.6 Conclusion

In this section, we propose a continual representation learning framework for sound classes. In this framework, the encoder training does not rely on labels and therefore is suitable for the practically relevant case where only a subset of labels is used to finetune an output classifier. Additionally, the framework is flexible enough to continually incorporate novel classes without the apriori knowledge of the total number of classes. With the continually pre-trained representations, the computational burden is significantly alleviated by simply finetuning a shallow classifier for downstream tasks. We showed, for the first time, that continual self-supervised learning gets competitive performance even if we do not use any mechanism against forgetting, which helps to reduce computational complexity. In future work, we plan to integrate continual self-supervised learning on more challenging audio tasks such as multi-label classification.

CHAPTER 5: SEMI-SUPERVISED SOUND CLASSIFICATION AND SEPARATION

Learning without human annotations has been a long-standing research problem. Audio data with high-quality human annotations are costly or even infeasible to obtain. For classification problems, audio clips may contain overlapping events, and it requires extensive efforts to verify all classes are correctly identified in the label. For source separation tasks, the clean reference signal for each individual source is required as the training target, but the source signal from real-world mixtures hardly exists in isolation, and synthetic mixtures with available source signals are commonly used as a resort. In contrast, audio data without reference exists at a large scale in the wild.

One way to leverage audio data without annotations is self-supervised learning. As described in Sections 2.3 and 4.2, low-dimensional representations are first learned through a pretraining stage using proxy labels that come from the data itself; then, the learned representations are applied to downstream tasks with a small set of labeled data that can be used to fine-tune the output head. Since label information is not used at all during the pretraining stage, task-specific performance is not guaranteed to be on par with the end-to-end supervised approaches. On the other hand, semi-supervised learning approaches, first introduced in Section 2.4, focus on incorporating a massive amount of unlabeled data into the joint training with a limited set of data with annotations. The bootstrapping framework is a popular semi-supervised approach, where a teacher model trained with labeled data is used for assigning pseudo-labels for data without annotations. These pseudo-labeled data will be used in conjunction with the labeled data to train a refined student model. To investigate semi-supervised learning in the audio domain, in Section 5.1, we integrate bootstrapping with contrastive learning in cross-modal audio-text representation learning where we improve the state-of-the-art approach by curating audio-text pairs from unaligned audio and text corpus. We further incorporate bootstrapping for the task of singing voice separation in Section 5.2. Aside from bootstrapping methods, in Section 5.3, we also consider regularization-based semi-supervised learning for personalized speech enhancement by designing separate loss functions for data with and without reference speech, making it possible to learn from mixtures without corresponding clean speech of the target speaker. With the effective usage of the in-the-wild data without reference, we empirically show that the proposed methods surpass conventional supervised approaches relying solely on human annotations.

[4]Part of this chapter has been published in [62, 63, 64].

5.1 SEMI-SUPERVISED IMPROVEMENT FOR AUDIO-TEXT CROSS-MODAL REPRESENTATIONS

5.1.1 Cross-Modal Representation Learning

Representation learning methods such as Self-Supervised Learning (SSL) [187] expand the limited scope of supervised learning by learning representations that can be applied to a large variety of downstream tasks. However, the mainstream SSL methods in the literature typically train an encoder on uni-modal data [42, 46, 50].

Learning cross-modal representations that involve text adds the additional flexibility of incorporating language in downstream tasks. This enables downstream tasks such as zero-shot classification possible, where the model is able to perform classification without being restricted by a pre-defined and explicitly annotated label set. The cross-modal representations can also be used in other tasks, such as audio-to-text and text-to-audio retrieval.

Learning cross-modal representations has been explored in computer vision under prior works including CLIP [188], Florence [189], and ALIGN [190]. AudioCLIP [191] extends the CLIP framework to the audio domain by incorporating an audio encoder to learn a joint embedding space for audio, vision, and language using aligned data across the three domains. CLAP [192], on the other hand, learns audio-text embeddings directly without depending on the image domain. It aims to maximize the similarity of the text-and-audio representations that correspond to the paired audio and caption within a given batch. To train semantically meaningful representations, these methods require a large number of paired audio and text items. While in-the-wild audio-text pairs exist at an extensive scale (e.g., captions from online video, metadata from audio datasets), the text is likely to be irrelevant to the sound events presented in the audio and is therefore unsuitable for training audio representations. Collecting high-quality captioned audio is an expensive task, hence data availability might constitute a bottleneck for scaling the audio-text pretraining. To overcome the limitation, Wav2CLIP [193] uses audio-image pairs from video clips to learn audio-text correspondence by distilling from the pre-trained CLIP model. VIP-ANT [194] extends Wav2CLIP by mining additional audio-text pairs from video and text data using pre-trained CLIP. The LAION-CLAP [195] augments the training data by performing keyword-to-caption generation from tags or labels of audio clips using a pre-trained language model. These approaches, however, assume the existence of either an additional anchor modality with paired annotations or a pre-trained model for generating text.

In this work, we explore the possibility of improving the zero-shot classification performance of audio-text representations using unpaired text and audio. For this purpose, we first train an initial teacher model using paired audio clips and captions; we then use this teacher model to automatically align textual descriptions to in-the-wild audio files, using the pairs aligned with higher confidence to train a new model. We then argue that this performance can be further improved by curating a refined, domain-specific dataset that is more akin to the zero-shot classification domain. Our

contributions are listed as follows,

- We propose domain-unspecific and domain-specific curation methods and show the improvement on three downstream zero-shot audio classification tasks.
- We propose a soft-labeled training objective that can avoid learning with hard labels in cases where the batch contain similar data items. We show that this training objective significantly improves the performance under domain-specific curation strategies.
- We show that the proposed curation methods significantly improve the model even in the case where the teacher model is trained on a small amount of paired data (%10 of the data).

5.1.2 Contrastive Audio-Text Pretraining

Training Framework

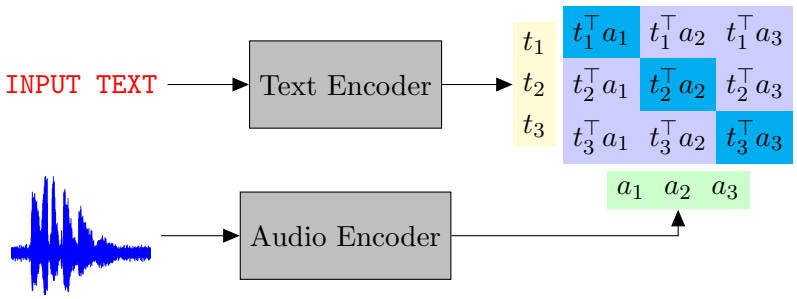

Figure 5.1: The training of the CLAP model for learning cross-modal representations. Input data from each domain is mapped to the shared latent space with the corresponding encoder network. The similarity between the embeddings of the audio-text pairs (diagonal terms of the similarity matrix, shown in blue) is maximized.

Our proposed method is based on CLAP, "Contrastive Language-Audio Pretraining" [192], which learns a joint latent space between text and audio by maximizing the similarity between the text and audio latent representation for the text caption and its corresponding audio signal. Let, X_t, X_a respectively denote a batch of text and audio. In the CLAP model, the latent representation is obtained by passing the text and audio through the text and audio encoders $f_t(.)$, and $f_a(.)$ such that,

$$L_t = f_t(X_t),\ L_a = f_a(X_a), \tag{5.1}$$

where $L_t \in \mathbb{R}^{N\times T}$, $L_a \in \mathbb{R}^{N\times A}$, such that T is the latent dimensionality of text, A is the latent dimensionality of audio, and N is the batch size. CLAP trains a joint latent space by passing L_t

and L_a through fully-connected layers such that,

$$t = \mathbf{FC}_t(L_t),\ a = \mathbf{FC}_a(L_a), \tag{5.2}$$

where $\mathbf{FC}(.)$ denotes the multi-layer perceptron transformation layers, $t \in \mathbb{R}^{N \times d}$, and $a \in \mathbb{R}^{N \times d}$ respectively denote the latent variables with same latent dimensionality d. The model then tries to maximize the diagonal entries on the matrix $C = ta^\top$. This translates into the following training loss function,

$$\mathcal{L}(C) = \frac{1}{2} \sum_{i=1}^{N} \Big(\log(\mathbf{Softmax}_t(C/\tau)_{i,i}) + \log(\mathbf{Softmax}_a(C/\tau)_{i,i}) \Big), \tag{5.3}$$

where $\mathbf{Softmax}_t(.)$ and $\mathbf{Softmax}_a(.)$ respectively denote Softmax functions along text and audio dimensions, τ is a learnable temperature scaling parameter, and the $C_{i,i}$ denotes the diagonal elements of the C matrix. We show the training forward pass pipeline in Figure 5.1.

Zero-shot Classification

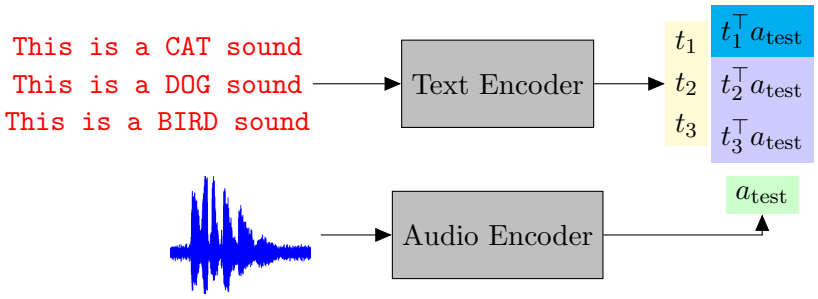

Figure 5.2: Zero shot classification with the CLAP model. Each class label is first prepended with a prompt to make it a caption, and the similarity between this caption's embedding and the test audio's embedding is computed. The class label associated with the caption that obtains the highest similarity score with the audio embedding is used as the predicted output.

The CLAP model is able to perform zero-shot classification by simply calculating the similarity of a given audio to a fixed set of text prompts constructed from class labels. That is, the classification decision is simply taken to be

$$\widehat{c} = \arg\max_j t_j^\top a_{\text{test}}, \tag{5.4}$$

where $\widehat{c}$ is the zero-shot classification decision, a_{test} is the embedding for the test audio, and t_j is the text embedding corresponding to the label of class j. We show the pipeline of zero-shot classification in Figure 5.2.

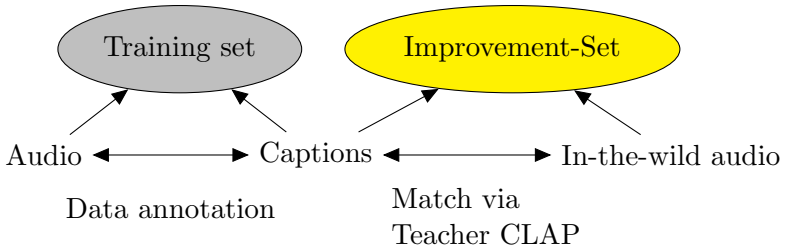

Figure 5.3: Domain-Unspecific (DU) Curation of Improvement-Set. With the pre-trained teacher CLAP model, the embedding of each in-the-wild audio clip is matched against the embeddings of all training captions. The new audio-text pairs with similarity above a threshold σ are included in the Improvement-Set.

5.1.3 Semi-supervised Improvement of Cross-Modal Audio-Text Representations

In this section, we describe the different strategies we employ to curate a paired dataset from unpaired text and audio. This curated dataset is used to train a student model in order to improve upon the zero-shot performance of the teacher model. We call this curated dataset the *Improvement-Set*. We follow two different strategies to curate this dataset. We call the first strategy the "Domain-Unspecific" (DU) dataset curation, where we form pairs using in-the-wild audio and text. The second strategy is "Domain-Specific" (DS) where we explicitly try to find audio recordings that are more relevant to the zero-shot classification task at hand.

Domain-Unspecific Improvement-Set Curation

For this strategy, we use a teacher CLAP model to curate an Improvement-Set. We match the captions of the training data with audio from a large dataset such as AudioSet [196]. We compute the cosine similarity between each pair of audio and text embeddings, and keep pairs with similarity above a threshold $\sigma \in [0, 1]$. We show this strategy in Figure 5.3. As we showcase in the experiments, this strategy provides an improvement over the base model; however, with a domain-specific refinement of the Improvement-Set, we can further enhance the zero-shot performance.

Domain-Specific Improvement-Set Curation

For this strategy, the main idea is to narrow down the captions used for the Improvement-Set by calculating text-to-text similarities between the captions from the training set and in-domain labels for a zero-shot classification task. Once these similarities are obtained, the first domain-specific (DS) strategy is to narrow down the training set by picking the subset of audio-text pairs that are most related to the downstream task through the text modality, determined by the maximum similarity between the embeddings of the caption and all the in-domain labels of the downstream task. We illustrate this procedure in Figure 5.4.

Another option is to augment the domain-specific Improvement-Set by involving in-the-wild

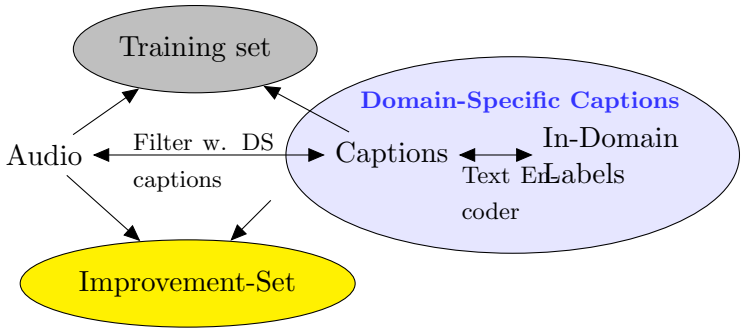

Figure 5.4: Domain-Specific (DS) Curation of Improvement-Set with Domain-Specific Audio. The Improvement-Set consists of audio-caption pairs from the training set that are most relevant to the downstream task by measuring the similarity between the embeddings of the training captions and in-domain labels of the downstream task.

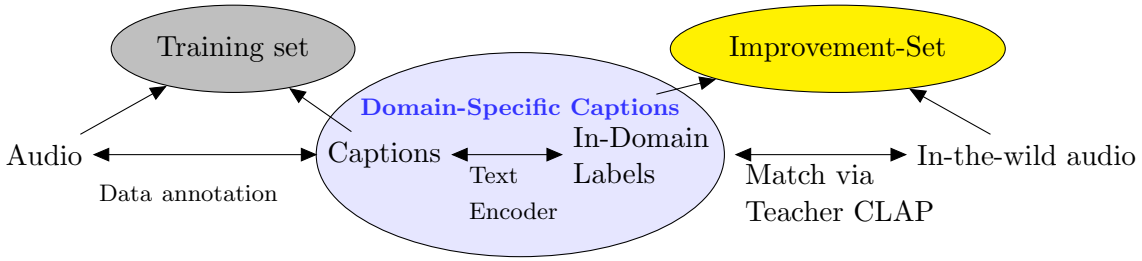

Figure 5.5: Augmented Domain-Specific (ADS) Curation of Improvement-Set. First, DS Curation is performed to obtain the subset of captions from the training set that are most related to the downstream task. Then, in-the-wild audio clips are aligned with this subset of captions with the pre-trained teacher CLAP model.

audio data. The process to create the Augmented Domain-Specific (ADS) Improvement-Set, as demonstrated in Figure 5.5, is as follows:

1. We calculate the similarities between the text embeddings of class labels from the in-domain dataset and the captions from the teacher's training set. We take the most similar captions from this set by thresholding the similarity values.
2. We find the correspondences between the domain-specific captions and the in-the-wild audio using the teacher CLAP.

5.1.4 Using Soft Labels in Contrastive Loss

An underlying issue with the original CLAP learning objective in Equation (5.3) is that it neglects local similarities for data within the same batch and treats all negative samples equally. It is possible to sample audio-text pairs with similar content from the same batch (i.e, multiple clips of "dog-barking"), while the objective in Equation (5.3) penalizes the model for not discriminating these similar instances. This issue is exacerbated when the training data is less diverse. For

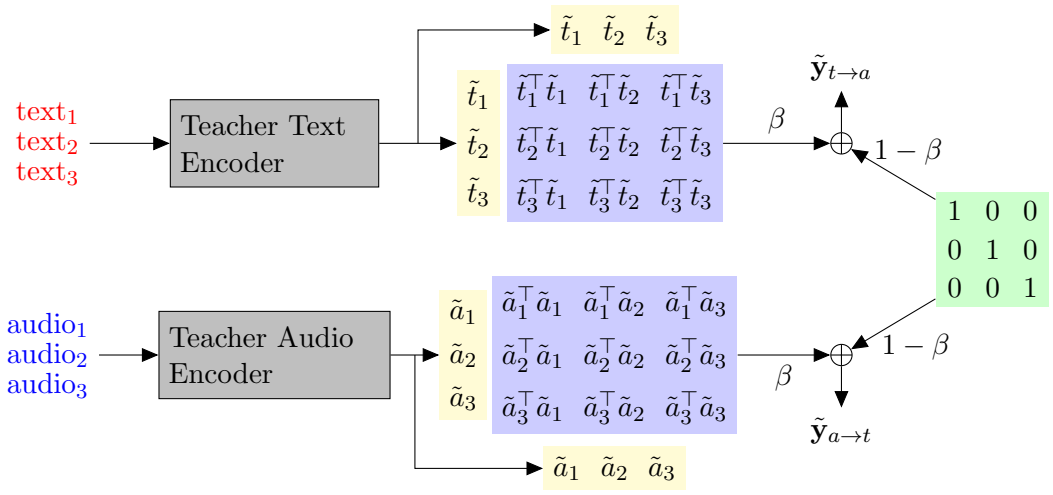

Figure 5.6: Computation pipeline for soft-labeled loss function. We first obtain the intra-batch similarity using the audio and text embeddings computed from the pre-trained teacher model. The soft labels are defined as the linear interpolation between the intra-batch similarity and the one-hot labels (rows or columns of the identity matrix).

instance, with the DS or ADS Improvement-Set, it is more likely to sample similar inputs within a batch.

The original learning framework of CLAP is equivalent to solving an N-class classification problem for a batch size of N, where the label for the i-th sample of each batch is the one-hot vector $\mathbf{y}_i \in \mathbb{R}^N$ with only the i-th entry equal to 1. Similar to prior studies in image-text pretraining [197, 198], we propose a label-softening technique when training the student model. The soft labels are obtained from the similarity between input samples within each batch, as illustrated in Figure 5.6. For each input batch, we obtain the soft targets by estimating the intra-modal similarity using the teacher audio ($\tilde{a}$) and text ($\tilde{t}$) embeddings. We define,

$$\tilde{C}_a = \tilde{a}\tilde{a}^\top, \ \tilde{C}_t = \tilde{t}\tilde{t}^\top, \tag{5.5}$$

where $\tilde{C}_a, \tilde{C}_t \in \mathbb{R}^{N\times N}$ represent the intra-batch, intra-modal similarity matrices for audio and text, respectively. For the i-th sample in this batch, we define the audio-to-text soft label as,

$$\tilde{\mathbf{y}}^{(i)}_{a\to t} = (1-\beta)\mathbf{y}_i + \beta\tilde{C}^{(i)}_a, \tag{5.6}$$

where $\tilde{C}^{(i)}_a$ is the i-th row of the matrix $\tilde{C}_a$ and the coefficient
$\beta \in [0, 1]$ adjusts the weights of the hard and soft labels; in symmetry, the text-to-audio soft label can be formulated as,

$$\tilde{\mathbf{y}}^{(i)}_{t\to a} = (1-\beta)\mathbf{y}_i + \beta\tilde{C}^{(i)}_t. \tag{5.7}$$

The overall learning objective is therefore,
$\tilde{\mathcal{L}}(C) = \frac{1}{2}(\tilde{\mathcal{L}}_t(C) + \tilde{\mathcal{L}}_a(C))$, with

$$\begin{aligned}\tilde{\mathcal{L}}_t(C) &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{D}_{KL}(\tilde{\mathbf{y}}_{a\to t}^{(i)} \| \mathbf{Softmax}_t(C/\tau)^{(i)}), \\ \tilde{\mathcal{L}}_a(C) &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{D}_{KL}(\tilde{\mathbf{y}}_{t\to a}^{(i)} \| \mathbf{Softmax}_a(C/\tau)^{(i)}),\end{aligned} \tag{5.8}$$

where $\mathcal{D}_{KL}$ denotes KL-Divergence, and C denotes the similarity matrix obtained from the student model.

5.1.5 Experimental Setup

Downstream Evaluation

We consider the following datasets for downstream evaluation with sound event classification and acoustic scene classification tasks:

- ESC-50 [66] with 2000, 5-second audio clips from 50 environmental sound classes.
- UrbanSound8K [182] with 8732, 4-second recordings from 10 possible urban sound classes.
- TUT Acoustic Scenes 2017 (TUT17) [199] with 6300, 10-second clips from 15 possible scene classes.

We measure the performance of the models using the accuracy from zero-shot evaluation described in Section 5.1.2. Following [192], we add a prefix prompt "*this is a sound of [class label]*" to each label in the downstream dataset before computing text embeddings.

Training Data

DU	**ESC-50**		**UrbanSound8K**		**TUT17**	
	DS	ADS	DS	ADS	DS	ADS
5.0×10^6	15.6×10^3	1.30×10^6	9.7×10^3	1.1×10^6	8.7×10^3	0.3×10^6

Table 5.1: Number of audio-caption pairs used in the curated Improvement-Set. DU represents the domain-unspecific curation; DS refers to the domain-specific curation containing the subset of the teacher training set relevant to each downstream task; ADS is the domain-specific curation from the in-the-wild audio.

For training the teacher CLAP, we follow the original paper [192] by using 128k paired audio and text captions from FSD50k [200], ClothoV2 [201], AudioCaps [202], and MACS [203]. For

the unsupervised curation in Section 5.1.3, we use the captions from the teacher training set and recordings from the unbalanced training set from AudioSet [196] by excluding recordings that are corrupted or contained in AudioCaps for the teacher training, which results in more than 1.9 million clips; notice that no label information from AudioSet is used during curation. We use a similarity threshold $\sigma = 0.7$ for the domain-unspecific curation and a threshold between 0.6 and 0.75 for the domain-specific experiments. The numbers of audio-caption pairs of the curated datasets are outlined in Table 5.1.

During training, each input audio recording is resampled to 44.1 kHz. If longer than 5 seconds, a random 5-second segment of the recording is chosen; if shorter, the recording is zero-padded.

Training Details

Following CLAP [192], we use the CNN14 [17] as the audio encoder and the BERT [42] as the text encoder, each followed by a two-layer MLP as the projection layer. The CNN14 is initialized from the AudioSet pre-trained checkpoints with a sampling rate of 32 kHz [17], and the BERT is initialized from the BERT base uncased in [204]. The embedding vectors have a dimension of 1024. The temperature parameter τ is initialized to 0.007, and the coefficient β for soft labels is set to 0.3. The student model is initialized from the teacher model. All models are trained with 2 Quadro RTX 6000 GPUs using a batch size of 64 per GPU. We perform three runs for each setup and obtain the average results.

For student training with the DU Improvement-Set, we randomly replace the batches with samples replayed from the teacher training set to combat catastrophic forgetting [53]. For student training with ADS Improvement-Set, the replay dataset is selected as the subset of audio-text pairs that are relevant to the downstream task, namely, the DS Improvement-Set. Note that with the addition of replay, we observed a performance gain of 0.6%, 0.9%, and 2.1% under the ADS curation strategy, on ESC-50, Urbansound8K, and TUT17, respectively compared with the models trained without using replay. We noticed similar trends with other curation strategies as well, so we used replay in all of our results.

5.1.6 Results and Discussions

Student Training from Full-dataset Teacher

In Table 5.2, we showcase the improvements obtained when the teacher model is trained with the full training set. We first observe that the teacher model with our training conditions can replicate the performance of the CLAP model released by Microsoft [192] (that we refer to as MS-CLAP). On UrbanSound8K, our implementation exceeds the performance of MS-CLAP by 1.4%. When further training the teacher CLAP model with the soft-labeled loss (OC+SL), the accuracy increases by 1.2% on ESC and by 0.3% on TUT17 while decreasing by 0.9% on UrbanSound8K

	Zero-Shot Evaluation Set		
Model	**ESC-50**	**UrbanSound8K**	**TUT17**
MS-CLAP	82.6	73.4	29.6
Our-CLAP (OC)	81.9 ± 0.9	74.8 ± 1.2	29.8 ± 1.3
OC+SL	83.1 ± 1.2	73.9 ± 2.6	30.1 ± 2.1
OC, DU	82.4 ± 1.4	73.9 ± 0.2	31.5 ± 1.0
OC, DU+SL	83.0 ± 0.5	74.9 ± 1.4	29.9 ± 1.9
OC, DS	78.8 ± 0.5	73.2 ± 1.1	29.8 ± 2.6
OC, DS+SL	83.5 ± 0.6	75.5 ± 1.4	31.8 ± 2.6
OC, ADS	84.2 ± 0.5	74.2 ± 2.1	32.5 ± 1.0
OC, ADS+SL	**85.1** ± 0.7	**77.4** ± 0.6	**36.0** ± 1.8

Table 5.2: Accuracy (percentage) on zero-shot evaluation for teacher and student models based on the teacher trained using full dataset. DU, DS, ADS, and SL denote Domain-Unspecific, Domain-Specific, Augmented Domain-Specific, and soft-labeled loss. Best performances are highlighted in bold.

compared to the teacher model, indicating that soft-labeled loss alone cannot yield significant and consistent performance gains.

We next observe that the domain-unspecific (DU) improvement of the CLAP model improves the zero-shot performance in certain cases. On ESC-50, especially with the addition of soft labels, we are able to increase the zero-shot accuracy from 81.9% to 83.0%. However, we do not observe an improvement for all three downstream datasets with DU.

When curating the Improvement-Set with the downstream domain knowledge, we observe that the Domain-Specific (DS) curation alone does not improve the performance: The results validate our hypothesis that learning with hard labels with similar input data from the same batch would adversely impact the quality of learned representations. However, we observe that adding the soft-labeled loss significantly boosts the zero-shot accuracy by exceeding the performance of the teacher model on all three datasets. We finally make the observation that the augmented domain-specific curation along with soft labels (ADS+SL) substantially improves the performance across the board. The results suggest that additional data with domain-specific curation and soft-labeled loss become more effective when used in conjunction, each of which is crucial to performance improvement.

Student Training from Subset Teacher

In prior experiments, we assume the availability of adequate paired audio and text data for training the teacher model. We would also like to investigate how data curation, label softening, and subsequent student training can be impacted if the amount of paired audio and text is limited during teacher training. To this end, we perform the teacher training by randomly sampling 10% of the original 128×10^3 paired audio and caption data. We follow the identical setup for the unsupervised curation and label softening as in the experiments involving the complete dataset.

	Zero-Shot Evaluation Set		
Model	**ESC-50**	**UrbanSound8K**	**TUT17**
CLAP teacher (full-dataset)	81.9 ± 0.9	74.8 ± 1.2	29.8 ± 1.3
CLAP teacher (subset)	74.2 ± 1.3	73.5 ± 2.0	30.9 ± 1.6
DU + SL	78.9 ± 0.3	73.7 ± 1.3	28.8 ± 1.1
ADS + SL	**81.3** ± 1.0	**74.5** ± 0.7	**31.3** ± 0.5

Table 5.3: Zero-shot accuracy for experiments where the teacher model is trained with 10% of the original training data. As a reference, we also include the CLAP teacher model trained with the full dataset from Table 5.2. The best subset model is highlighted in bold.

We outline the zero-shot accuracy of the teacher and student models in Table 5.3.

Student models trained with the DU Improvement-Set enhance the performance on ESC-50 (from 74.2% to 78.9%) and UrbanSound8K (from 73.5% to 73.7%) while slightly degrading on TUT17 (from 30.9% to 28.8%); the results can be attributed to the domain-unspecific curation, which yields data that are more relevant to environmental sound classes and distinct from acoustic scene recordings. By incorporating domain-specific knowledge, the ADS strategy further enhances the performance of ESC-50 (to 81.3%) and UrbanSound8K (to 74.5%) beyond that of the teacher model and also exhibits a slight improvement on TUT17 (to 31.3%). Quite notably, the performance on ESC-50 and UrbanSound8K is similar to that of the teacher model trained with the full dataset. These improvements observed in the student training demonstrate the effectiveness of the data curation and label softening techniques proposed, even in the absence of ample paired multi-modal data for training the teacher model.

5.1.7 Conclusion

In this work, we propose domain-unspecific and domain-specific data curation methods to obtain additional audio-text pairs from unaligned audio and text corpus for audio-text pretraining. The proposed methods can effectively improve the zero-shot classification performance of cross-modal representations. We have identified that using domain-specific dataset curation combined with a soft-labeled loss significantly improves the performance over a baseline teacher model. This observation holds even in the case where the baseline teacher is trained with 10% of the original training data. We plan to further improve our approach by incorporating general text corpora into the curation pipeline and exploring more advanced algorithms for audio-text matching in future work.

5.2 SEMI-SUPERVISED SINGING VOICE SEPARATION WITH NOISY SELF-TRAINING

5.2.1 Supervised and Semi-Supervised Learning for Singing Voice Separation

The task of singing voice separation is to separate the input mixture into different components: singing voice and accompaniment. It is a crucial problem in music information retrieval and has commercial usage such as music remixing and karaoke applications. It also has the potential to provide useful information for downstream tasks such as song identification, lyric transcription, singing voice synthesis, and voice cloning without access to clean sources.

Deep learning models have recently shown promising results in singing voice separation. Popular methods are mostly supervised methods, where a deep neural network is trained on a multi-track corpus with paired vocal and accompaniment ground-truths. A common approach is to apply dense connections between convolutional or long short-term memory (LSTM) blocks to estimate separate masks [205, 206], or to use a bidirectional LSTM (BLSTM) network in the separator [207]. Models with multi-scale processing further improve the performance of separation. With the concatenation of features at different scales along with skip connections, U-Net [208] can maintain long-term temporal correlation while processing local information with higher resolution. Such architecture has been effective in both time-frequency domain [209, 210, 211, 212] and end-to-end, time-domain methods [213, 214]. Inspired by [104] in monaural speech separation, models that simultaneously process features at different resolutions with multiple paths have also shown effectiveness in singing voice separation systems [215, 216].

The primary challenge for supervised methods with deep learning is the lack of training data with ground-truth. It is more significant for larger networks that are more prone to overfitting issues. There are several multi-track datasets publicly available for singing voice separation including MIR-1K [217], ccMixter [218], and MUSDB [219]. However, these datasets are relatively small (all these combined are around 15 hours) and not diverse. To artificially increase the size of the dataset, data augmentation methods, including random channel swapping, amplitude scaling, remixing sources from different songs, time-stretching, pitch shifting, and filtering, are applied to the input signal [210, 220, 221]. These methods, individually or combined, are empirically shown to enhance separation performance only by a limited margin [210].

On the other hand, semi-supervised and unsupervised methods do not require a large corpus with a one-to-one correspondence between the mixtures and ground-truth sources. Demucs [213] leverages mixture data by first training a silent-source detector on a small labeled dataset, then mixing recordings with only one source and mixture recordings with the source being silent, and finally optimizing with a weakly supervised loss. Methods based on generative adversarial networks [222, 223] require only the isolated sources for training. The distance between the distributions of separator's output and the isolated sources is minimized with adversarial training. In [224, 225], unpaired vocal and accompaniment data are used to learn non-negative, smooth representations with a denoising auto-encoder using an unsupervised objective. Authors for [226] propose a stage-

wise algorithm where a clustering-based labeler assigns time-frequency bin labels with confidence measure, and a student separator network is trained on these labels afterward.

Self-training is a semi-supervised framework in which a pre-trained teacher model assigns pseudo-labels for unlabeled data. Then a student model is trained with the self-labeled dataset. It has been applied in several applications such as image recognition [57] and automatic speech recognition [61, 227]. Our approach follows the noisy self-training method in which we investigate data augmentation methods on musical signal and evaluate how they affect the separation performance.

With the framework of noisy self-training, we aim to improve the performance of a deep separator network where only a limited amount of data with ground-truth is available. The contribution of this work is listed as follows:

- We use a large unlabeled corpus to improve separation results under the noisy self-training framework.
- We show how data augmentation can improve the model's ability to generalize with a focus on random remixing between sources.
- We propose to use a voice activity detector to evaluate the quality of self-labeled data in the student training to perform data filtering.

5.2.2 Noisy Self-Training for Singing Voice Separation

Overview of Pipeline

We propose a noisy self-training pipeline based on the concept of self-training introduced in Section 2.4. Our proposed self-training framework for singing voice separation consists of the following steps:

1. Train a teacher separator network $f^{(0)}$ on a small labeled dataset $\mathcal{D}_l$.
2. Assign pseudo-labels for the large unlabeled dataset $\mathcal{D}_u$ with $f^{(0)}$ to obtain the self-labeled dataset $\mathcal{D}^{(0)}$.
3. Filter data samples from $\mathcal{D}^{(0)}$ to obtain a refined self-labeled dataset $\mathcal{D}_f^{(0)}$.
4. Train a student network $f^{(1)}$ with $\mathcal{D}_l \cup \mathcal{D}_f^{(0)}$.

This framework can be made iterative by repeating steps 2 to 4, using the student network $f^{(i)}$ as the new teacher to obtain a self-labeled dataset $\mathcal{D}^{(i+1)}$, filter low-quality samples to obtain $\mathcal{D}_f^{(i+1)}$, and training a new student model $f^{(i+1)}$. The process stops when there is no performance gain. We illustrate the framework pipeline in Figure 5.7.

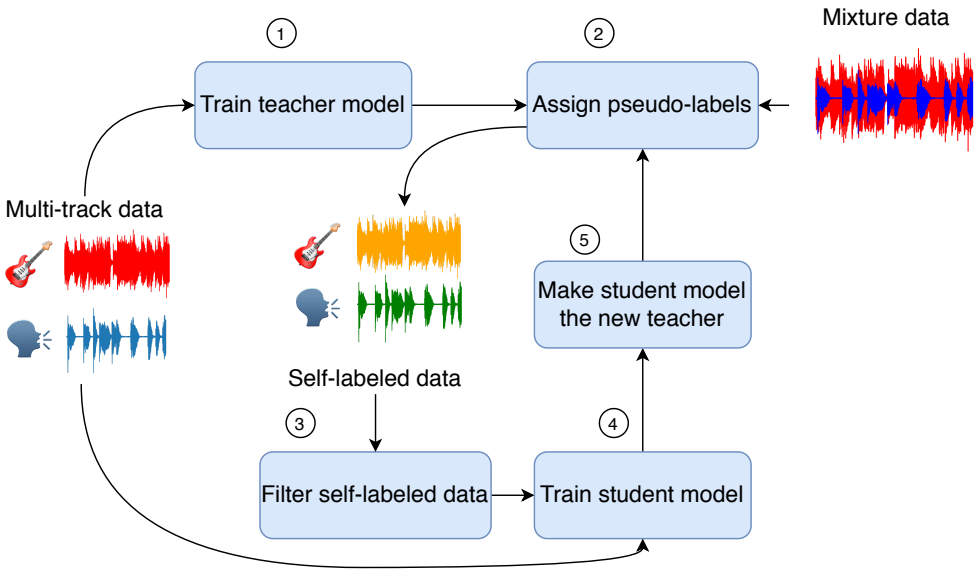

Figure 5.7: The pipeline of noisy self-training for singing voice separation.

Data Filtering with Voice Activity Detector

The primary difference between the proposed self-training framework and conventional bootstrapping setups with teacher-student training in Section 2.4 is the data filtering step. For singing voice separation tasks, poor quality self-labeled samples may contain leakage of the singing voice in the accompaniment tracks or leakage of musical background in the vocal tracks. To filter out these samples, we evaluate the quality of the data with a voice activity detector (VAD).

The VAD takes the STFT magnitude spectrogram of the mixture as input and predicts the frame-level energy ratio between the source and the mixture. We use the 2D-CRNN architecture with the same configuration as in [228]. We train two separate VADs to estimate the energy ratio of vocal over mixture, and accompaniment over mixture, respectively. The ground-truth is defined as 0 when both the vocal and the accompaniment are silent. The VADs are trained with the same labeled dataset as the one for the teacher separator model using binary cross-entropy loss.

To measure the leakage of accompaniment in vocal tracks, we pass the self-labeled vocal track into the accompaniment activity detector. Similarly, we feed the self-labeled background track into the vocal activity detector to detect leakage of the singing voice. A frame is defined as a "poor quality frame" if either its accompaniment energy in the vocal track or its vocal energy in the background track is higher than some threshold. We count the total number of "poor quality frames" for each song, and songs with a smaller percentage of such frames are considered to have higher quality.

Data Augmentation

Data noise is a key component in the noisy self-training framework. We apply data augmentation methods for the training of both teacher and student models. Each training sample contains both vocal and accompaniment tracks of 30 seconds duration. To augment the training set, we randomly select a window of duration T seconds (with $T < 30$) from the sample. We also perform "random mixing" by mixing vocal and background sources from two randomly selected songs with a probability of p. Besides, we apply dynamic mixing ratio, pitch shifting, lowpass filtering, and EQ filtering to the data.

5.2.3 Singing Voice Separator

Model Architecture

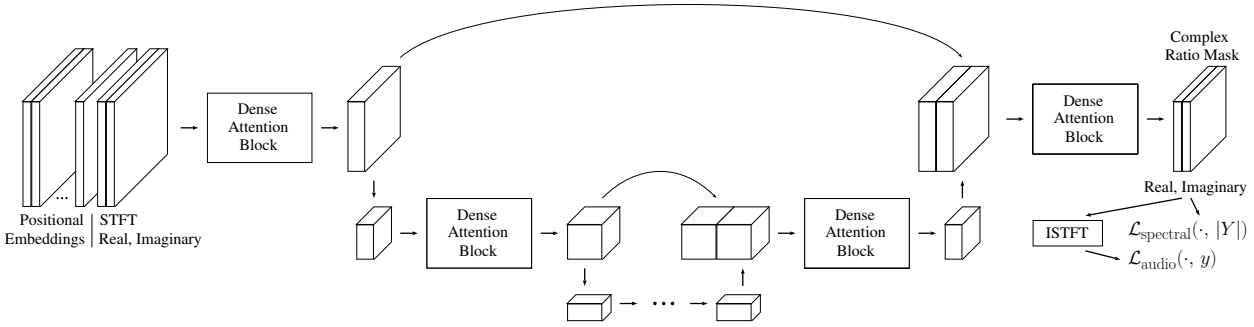

Figure 5.8: Top two levels of the U-Net architecture. Frequency positional embeddings are concatenated to the STFT real and imaginary parts as input to the network.

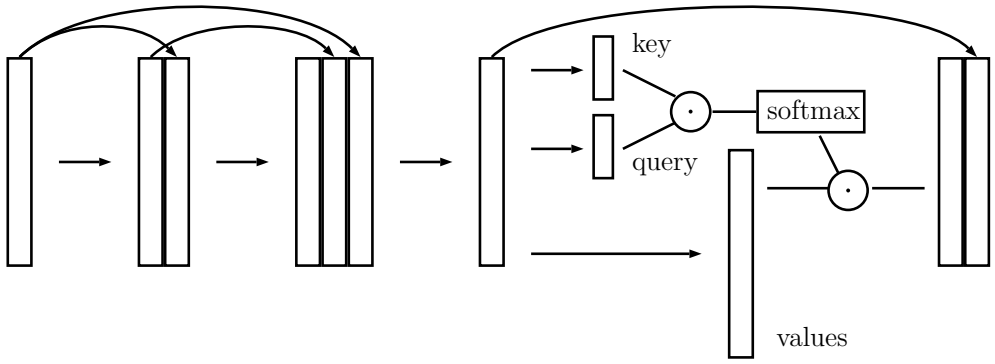

Figure 5.9: Details of the DenseNet and attention blocks. Straight arrows are convolutions with batch normalization and ReLU activation; curved arrows are concatenations.

We use the PoCoNet [101] for both teacher and student models. The neural network takes the concatenation of real and imaginary parts of the mixture's STFT spectrogram as input. The

separator estimates the complex ratio masks for each source. The wave-form signal is obtained by applying inverse STFT transform on the estimated spectrograms.

As shown in Figure 5.8, the separator is a fully-convolutional 2D U-Net architecture with DenseNet and attention blocks. The connection of layers in DenseNet and attention blocks is illustrated in Figure 5.9. Each DenseNet block contains three convolutional layers, each followed by batch normalization and Rectified Linear Unit (ReLU). Convolutional operations are causal only in the time direction but not in the frequency direction. We choose a kernel size of 3×3 and a stride size of 1, and the number of channels increases from 32, 64, 128 to 256. We control the size of the network by varying the number of levels in U-Net and the maximum number of channels. In the attention module, the number of channels is set to 5 and the encoding dimension for key and query is 20. The connections of layers in DenseNet and attention blocks follow [101]. Frequency-positional embeddings are applied to each time-frequency bin of the input spectrogram. For time frame t and frequency bin f, the embedding vector is defined as:

$$\rho(t, f) = (\cos(\pi \frac{f}{F}), \cos(2\pi \frac{f}{F}), \ldots, \cos(2^{k-1}\pi \frac{f}{F})), \tag{5.9}$$

where F is the frequency bandwidth and $k = 10$ is the dimension of the embedding.

Loss Functions

For each output source, the loss function is the weighted sum of wave-form and spectral loss:

$$\mathcal{L}_c(\mathbf{y}, \hat{\mathbf{y}}) = \lambda_{\text{aud}} \mathcal{L}_{\text{aud}}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda_{\text{spec}} \mathcal{L}_{\text{spec}}(\mathbf{Y}, \hat{\mathbf{Y}}), \tag{5.10}$$

where $c \in \{\text{voc}, \text{acc}\}$ denotes either the vocal or accompaniment source, $\mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^T$ are time-domain output and reference signals with length T, and $\mathbf{Y} = |\text{STFT}(\mathbf{y})|, \hat{\mathbf{Y}} = |\text{STFT}(\hat{\mathbf{y}})| \in \mathbb{R}^{F \times L}$ are the corresponding STFT magnitude spectrograms with F frequency bins and L frames. The time-domain and spectral-domain loss functions are weighted by λ_{aud} and λ_{spec}, respectively. We choose both $\mathcal{L}_{\text{aud}}(\cdot)$ and $\mathcal{L}_{\text{spec}}(\cdot)$ to be $\ell 1$ loss. The total loss is the weighted sum of each source:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \lambda_{\text{voc}} \mathcal{L}_{\text{voc}}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda_{\text{acc}} \mathcal{L}_{\text{acc}}(\mathbf{Y}, \hat{\mathbf{Y}}), \tag{5.11}$$

with the weighting coefficients λ_{voc} for vocal tracks and λ_{acc} for accompaniment tracks.

5.2.4 Experimental Setup

Data Preparation

We use MIR-1K [217], ccMixter [218], and the training partition of MUSDB [219] as the labeled dataset for supervised training. The training set contains approximately 11 hours of recordings.

We use DAMP [229] as the unlabeled dataset for training the student model. DAMP dataset contains more than 300 hours of vocal and background recordings from karaoke app users. Since these recordings are not professionally produced, there exists bleeding of music in the vocal tracks and bleeding of the singing voice in the accompaniment tracks as well; hence, it is not suitable for supervised source separation.

To reduce dimensionality and speed up processing, we downsample each track to 16 kHz and convert it to mono. We further segment the recordings to non-overlapping 30-second segments, and if the segment is less than 30 seconds we zero-pad to the end of the signal. The spectrograms are computed with a 1024-point STFT with a hop size of 256.

Noisy Self-Training Procedure

For both teacher and student training, we minimize Equation (5.11) with an Adam optimizer with an initial learning rate of 1e-4, and we decrease the learning rate by half for every 100k iterations until it's no greater than 1e-6. We set $\lambda_{\text{aud}} = \lambda_{\text{spec}} = 1$ in Equation (5.10) and $\lambda_{\text{voc}} = \lambda_{\text{acc}} = 1$ in Equation (5.11).

To augment the training set, we randomly select a window size of $T = 2.5, 5, 10$ seconds as the input to the model to experiment with the effect of input length, each with a batch size of 4, 2, and 1, respectively. The maximal batch size is chosen under the memory limit. We experiment with different probabilities of applying random mixing with $p = 0, 0.25, 0.5, 0.75, 1$.

The teacher model is trained on the labeled datasets. Then, it assigns pseudo-labels for the unlabeled dataset. We infer vocal labels using DAMP vocal tracks as input to the teacher model and infer accompaniment labels from DAMP accompaniment tracks. Due to the leakage in these vocal and background tracks, they can be viewed as mixtures where one source is more likely to dominate the other, compared to normal mixtures.

Evaluation Details

As in previous studies on singing voice separation [205, 206, 208, 211, 223], we measure the signal-to-distortion ratio (SDR) to evaluate the separation performance. Following the SiSec separation campaign [85], we use the 50 songs from the test partition of MUSDB [219] as the test set. We partition each audio track into non-overlapping one-second segments, and we take the median of segment-wise SDR for each song and report the median from all 50 songs. We use the python package `museval`[5] to compute SDR.

Len (s)	Size (1e6)	Prob RM	Use DAMP	SDR(V)	SDR(A)	Mean
2.5	8.3	0	No	1.84	10.31	6.08
		0.5		1.72	9.51	5.62
5		0		3.55	10.91	7.23
		0.5		4.08	11.34	7.71
10		0	Yes	3.93	11.46	7.70
		0	No	5.88	12.52	9.2
		0.25		6.35	12.56	9.46
		0.5		7.06	13.35	10.21
		0.75		6.98	13.36	10.17
		1.0		6.91	13.66	**10.29**
	1.6	0	Yes	0.03	6.62	3.33
		0	No	4.17	10.86	7.52
		0.5		4.34	11.13	7.74
	15.4	0	No	5.81	11.94	8.88
		0.5		6.9	13.07	9.99

Table 5.4: Test performance metrics (SDR in dB) for teacher model candidates. We experiment with various input sizes (Len), number of model parameters (Size), and the probability of random mixing (Prob RM) to pick the best configuration for the teacher model. The notations "V" and "A" denote vocal and accompaniment, respectively. The best performance is highlighted in bold.

5.2.5 Results and Discussions

Teacher Training

We select the configuration for the teacher model by experimenting with different input window sizes of training samples and the number of model parameters. Table 5.4 shows the test SDR for the combinations of input and model size. We first observe that using a longer input size improves both vocal and accompaniment SDR. The improvement can be attributed to the attention blocks where a longer input context provides more information for separation. Another observation is that larger models do not guarantee performance gain. The largest model (15.4M param) performs significantly better than the smallest one (1.6M) but is slightly worse than the 8.3M version for the probability of random mixing $p = 0, 0.5$. Using the best combination of input length (10 seconds) and model size (8.3M), we experiment with different probabilities of applying random mixing. As shown in [210], random mixing does not have a positive effect on test SDR, and one possible explanation is that it creates mixtures with somewhat independent sources. Our experiments, however, indicate that random mixing alone significantly improves the results. The best performance is obtained when random mixing is always applied. Our observations are consistent with the argument in [230] that

[5] https://sigsep.github.io/sigsep-mus-eval/

"one-versus-all" separation benefits from mixing independent tracks. Intuitively, mixtures with dependent sources are more difficult to separate. Random mixing makes it easier for the model to learn and to converge faster on the training set. Meanwhile, by mixing up sources from different songs, the training set becomes more diverse and the model has a better ability to generalize at inference time.

In addition, we verify that the DAMP dataset should not be applied directly in supervised source separation tasks by including this dataset along with the other labeled datasets. We experiment with two model sizes (1.6M and 8.3M) using 10-second input without random mixing, and the SDR values degrade sharply for both cases.

To summarize, we select the model using 10-second input window with 8.3M parameters and probability $p = 1.0$ of applying random mixing as the teacher model to infer pseudo-labels.

Student Training

Size (1e6)	top %	SDR(V)	SDR(A)	Mean
8.3	1	6.57	12.92	9.75
15.4	1	7.27	13.73	10.5
	0.5	7.52	13.91	10.72
	0.25	7.8	13.92	**10.86**

Table 5.5: Test performance metrics (SDR in dB) for student models. We experiment with different model sizes and the proportion of quality-controlled self-labeled samples (top %). The best performance is shown in bold.

Table 5.5 summarizes the test SDR for student models. As opposed to the teacher model, the 15.4M model has a 0.75 dB SDR gain compared to the 8.3M model. The observation that the larger capacity student model improves performance is consistent with the findings in [57].

To verify the quality control approach with VADs, we first count for each source track the number of "poor-quality frames" as defined in Section 5.2.2 from three different datasets: DAMP, self-labeled DAMP, and MUSDB. The input to the VADs is a 10-second clip, equivalent to 625 frames. From the visualization in Figure 5.10, the unprocessed DAMP contains the highest percentage of data with a large number of poor-quality frames, the distribution of MUSDB is concentrated in the low count region, while the self-labeled dataset lies in between. This implies that the count of "poor-quality frames" based on the output of VADs is a reasonable indicator of the quality of data samples. The experimental results demonstrate that the proposed data filtering method with VADs further improves the performance. The highest SDR is obtained when only the top quarter of the self-labeled data is included in the training. Incorporating a higher percentage of self-labeled data may provide more diversity but is more likely to include samples with poor quality, thus negatively affecting the model's performance.

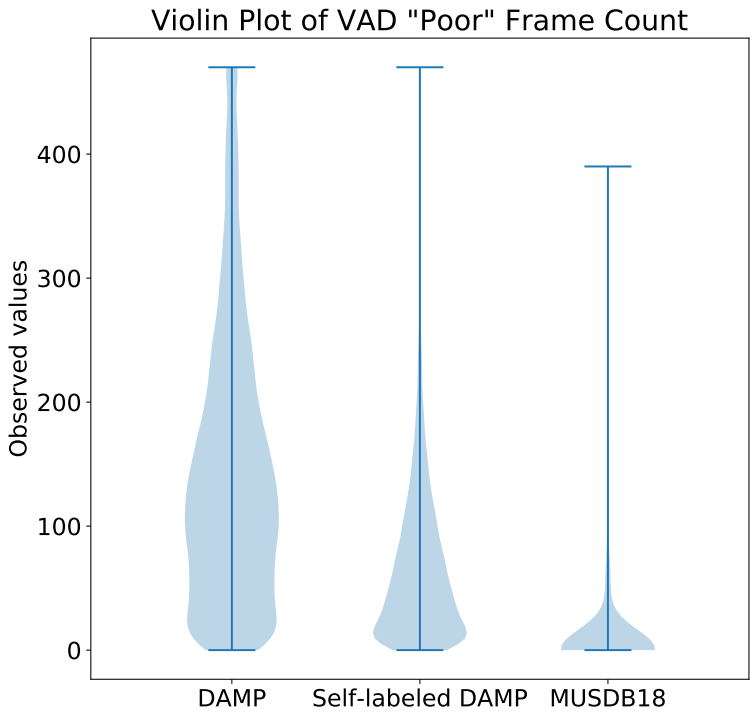

Figure 5.10: Count of "poor-quality frames" using reference vocal or accompaniment tracks for different datasets. Each input contains a total number of 625 frames. The higher the number of "poor-quality frames", the more likely there is leakage of vocal or accompaniment from the reference track.

Comparison with Other Methods

To compare the separation of singing voice with state-of-the-art, we also include models that separate the mixture into four sources, listed in Table 5.6. It has been shown in [210] that, these four-source models have similar vocal separation performance compared to two-source models, even though the four-source separation task is more challenging than the two-source counterpart; possibly because of the additional supervision provided by different instrumental sources in the multi-task learning setup. Hence, we include the vocal SDR values of state-of-the-arts for four-source models [214, 215] in our comparison. Our proposed approach, the student model using quality control with VADs, obtains the highest vocal and average SDR among all models, and the vocal separation outperforms others by a significant margin. The accompaniment SDR is higher than the baseline model with mono input [211] but worse than the stereo ones [205, 206]. Stereo input contains more spatial information for accompaniment than vocal since the left and right channel differences for background tracks are at a much larger scale than vocal tracks. Such information may improve the separation of accompaniment.

Name	#Src	Input type	Extra Data	SDR(V)	SDR(A)	Mean
Demucs[214]	4	Stereo	Labeled	7.05	N/A	N/A
Nachmani et. al. [215]			✗	6.92	N/A	N/A
MMDenseLSTM[206]	2	Stereo	✗	4.94	**16.4**	10.67
MDN[205]			✗	3.87	15.41	9.64
MT U-Net[211]		Mono	✗	5.28	13.04	9.16
Michelashvili et. al. [223]			✗	3.5	N/A	N/A
Ours (teacher)	2	Mono	✗	6.91	13.66	10.29
Ours (student, no VAD)			DAMP	7.27	13.73	10.5
Ours (student, VAD)			DAMP	**7.8**	13.92	**10.86**

Table 5.6: Test performance (SDR in dB) of the proposed method and other baseline models. We include models that take different numbers of input channels (Input type) and output channels (#Src) and list any extra dataset used for training the model (Extra Data) if applicable. The best performance is shown in bold.

5.2.6 Conclusion

We present a semi-supervised method for singing voice separation to deal with the scarcity of data with ground-truth. Using the noisy self-training framework, we can effectively make use of a large unlabeled dataset to train a deep separation network. Experimental results show that random mixing as data augmentation improves model training, and the data filtering method with pre-trained voice activity detectors improves the quality of the self-labeled training samples. Our study serves as a foundation for more complicated systems such as using stereo input, working with unlabeled datasets with mixture only (as opposed to noisy source tracks), and extending the teacher-student loop with additional iterations.

5.3 SEMI-SUPERVISED TIME-DOMAIN PERSONALIZED SPEECH ENHANCEMENT

5.3.1 Introduction to Personalized Speech Enhancement

Communications in the real world often take place in complex auditory scenes, where the speech of a target speaker is likely to be contaminated by an interfering speaker or background noise. To improve the quality of speech, deep learning methods have been developed for personalized speech enhancement (PSE), also known as target speaker extraction, with promising performance [231, 232]. PSE is distinct from the conventional speech enhancement (SE) task since the former requires the identification of a particular speaker while the latter is speaker-agnostic. As opposed to blind source separation (BSS) systems where all individual sources are separated by the model, PSE systems isolate the speech signal of the speaker in interest from all other speakers and ambient noise. By focusing only on the target speaker and treating the rest of the signal as interference,

PSE systems do not require information on the total number of sources beforehand and do not suffer from permutation ambiguity, which are the two primary challenges in BSS [103, 107].

The VoiceFilter [231] is one of the first neural-based architectures to extract the speech of a target speaker. It contains a bidirectional LSTM (BLSTM)-based separation module to extract the magnitude mask of the target signal from the mixture spectrogram. A pre-trained speaker embedding module extracts cues from the target speaker, which requires an enrollment utterance of the target speaker as an additional input. The speaker embedding guides the separator during the mask estimation. Several studies follow this embedder-separator design with improvements in the architecture to enhance the enhancement quality or running-time efficiency using spectral [232, 233, 234] or perceptual [67] features as input.

The recent success of end-to-end networks such as ConvTasNet [26] in BSS has drawn attention to time-domain architectures in PSE. The separator includes paired 1D convolutional encoder and decoder, along with stacks of convolutional blocks in the middle. The embedder network can be pre-trained [235] or learned jointly with the separator [236, 237, 238, 239, 240, 241]. Inspired by the Sepformer [105] in BSS, we propose the Exformer, a transformer-based network that can extract the speech of a given target speaker, and study the effectiveness of the multi-head attention (MHA) mechanisms with the transformer encoder [7] as the building block of the separator module for time-domain PSE. Specifically, we compare multiple configurations of the fusion of the speaker embedding into the separator module within the transformer blocks.

Despite the performance gain from the architectural improvement, these networks usually require a vast amount of labeled data to train. While it is unrealistic to obtain both the mixture and its source components in real-world acoustic conditions, noisy speech recordings exist on a large scale and are easy to collect. In the Mixture Invariant Training (MixIT) [97], separation models are trained with mixture of mixtures in both unsupervised and semi-supervised setups. An alternative approach to leverage the noisy mixtures is to train the network with pseudo-labels assigned by a pre-trained teacher model, with promising results in singing-voice separation [63, 242] and speech enhancement [243, 244]. Meanwhile, training with noisy data in PSE is yet to be explored. We propose a semi-supervised framework to train the Exformer with both labeled and unlabeled data. Empirical studies on two-speaker mixtures show that the proposed Exformer outperforms the prior architectures based on ConvTasNet under supervised learning, and the incorporation of unlabeled data with the semi-supervised setup further improves the quality of the extracted signal.

5.3.2 Model Architecture for Time-Domain Personalized Speech Enhancement

The proposed Exformer model follows the design of the masking-based audio source separation methods introduced in Section 2.2.2 with an encoder, a separator, and a decoder. As illustrated in Figure 5.11, Exformer is based upon the Sepformer [105] for time-domain speech separation. Apart from the masking-based separation components, a pre-trained BLSTM speaker embedder is used to extract speaker information to guide the extraction. The speaker extractor (encoder,

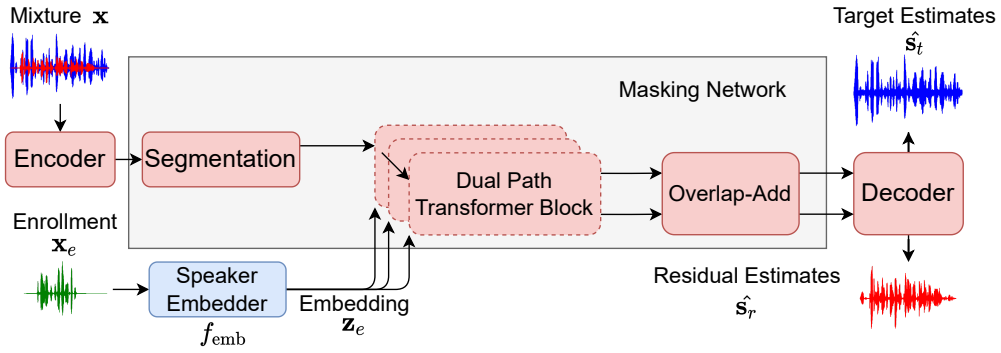

Figure 5.11: The architecture of the Exformer. The speaker embedder is pre-trained, and its weights are fixed during the training of the separation network. Speaker embeddings are applied to each of the dual-path transformer blocks.

masking network, and decoder) is trained with the negative scale-invariant signal-to-distortion ratio (SI-SDR) [106], and the parameters of the embedder network are fixed.

Speaker Embedder

The embedder network obtains a fix-sized embedding vector $\mathbf{z}_e = f_{\text{emb}}(\mathbf{x}_e) \in \mathbb{R}^D$ of the target speaker, where f_{emb} is the embedder network and $\mathbf{x}_e$ is the enrollment utterance. Following the embedder design in [231], the mel-spectrogram of the enrollment speech is processed by a stack of BLSTM layers to model sequential information. The average of the hidden-state representation from the first and the last frames is projected to a lower dimension with a linear layer. Finally, the projection output is normalized with unit $\ell 2$ norm to obtain the speaker embedding $\mathbf{z}_e$. The embedder network is pre-trained with the generalized end-to-end loss (GE2E) [245], which pushes the embedding to the centroid of all speech embeddings of the same speaker from the batch and away from the embeddings of other speakers.

Encoder

The encoder network consists of a one-dimensional convolutional layer followed by ReLU activation. It transforms the time domain input mixture $\mathbf{x} \in \mathbb{R}^L$ with L samples to a hidden representation $\mathbf{H} \in \mathbb{R}^{F \times T}$ with F features and T temporal frames.

Masking Network

The masking network takes the mixture representation $\mathbf{H}$ and the speaker embedding $\mathbf{z}_e$ as input and estimates a mask $\mathbf{M}_c \in \mathbb{R}^{F \times T}, c \in \{1, 2, \dots C\}$, where C is the total number of sources. We set $C = 2$ as the model estimates one mask for the target speech and another mask for the residual

interfering signal. Similar to DPRNN [104] and Sepformer [105], the masking network consists of three modules: segmentation, embedding-guided dual-path processing, and overlap-add.

Segmentation The segmentation module divides the T frames from the mixture representation $\mathbf{H}$ into S segments, each with length K, with 50% overlap between segments. All segments are concatenated to form a tensor $\mathbf{V}^{(0)} = \{\mathbf{V}_s^{(0)}\}_{s=1}^{S} \in \mathbb{R}^{F\times K\times S}$.

Embedding Fusion The embedding fusion module combines the segmented mixture representation and the target speaker embedding $\mathbf{z}_e$. We apply embedding fusion at the beginning of each processing block as in [237, 238, 239, 240]. At the beginning of block j, we perform

$$\mathbf{W}^{(j)} = f_{\text{fus}}^{(j)}(\mathbf{V}^{(j-1)}, \mathbf{z}_e), \tag{5.12}$$

where $\mathbf{W}^{(j)} \in \mathbb{R}^{F\times T\times S}$ is the embedding-guided mixture representation, $f_{\text{fus}}^{(j)}$ is the fusion function that applies independently to each segment and frame, and the superscript j denotes the block index.

We experiment with multiple approaches to incorporate speaker information into the separation pipeline. The first method concatenates the embedding and the latent mixture representation similar to [231, 237, 239, 240, 241, 246]. The embedding $\mathbf{z}_e$ is first duplicated on the frame and the chunk dimensions to obtain $\mathbf{Z}_e \in \mathbb{R}^{D\times K\times S}$. Then, $\mathbf{Z}_e$ is concatenated along the feature dimension to $\mathbf{V}^{(j-1)}$ and linearly projected back to the F-dimensional space:

$$f_{\text{cat}}^{(j)}(\mathbf{V}^{(j-1)}, \mathbf{Z}_e) = \text{Linear}(\text{Concat}(\mathbf{V}^{(j-1)}, \mathbf{Z}_e)). \tag{5.13}$$

In the second approach, we first project the embedding $\mathbf{Z}_e$ to the same dimension as $\mathbf{V}^{(j-1)}$ and then element-wise multiply with the mixture representation following [238]:

$$f_{\text{mult}}^{(j)}(\mathbf{V}^{(j-1)}, \mathbf{Z}_e) = \text{Linear}(\mathbf{Z}_e) \odot \mathbf{V}^{(j-1)}, \tag{5.14}$$

where $\odot$ stands for element-wise multiplication.

We also consider replacing the element-wise multiplication with addition similar to the application of positional embeddings in transformer [7], leading to our third approach:

$$f_{\text{add}}^{(j)}(\mathbf{V}^{(j-1)}, \mathbf{Z}_e) = \text{Linear}(\mathbf{Z}_e) + \mathbf{V}^{(j-1)}. \tag{5.15}$$

Dual-Path Processing Figure 5.12 (Left) shows the pipeline of a dual-path transformer block. After the embedding fusion, the dual-path module alternates the processing between short-term and long-term information. The intra-chunk layer computes the local dependency within each segment, while the inter-chunk layer communicates between the segments to model global information. As shown in Figure 5.12 (Right), each of the intra-chunk and inter-chunk layers contains multiple

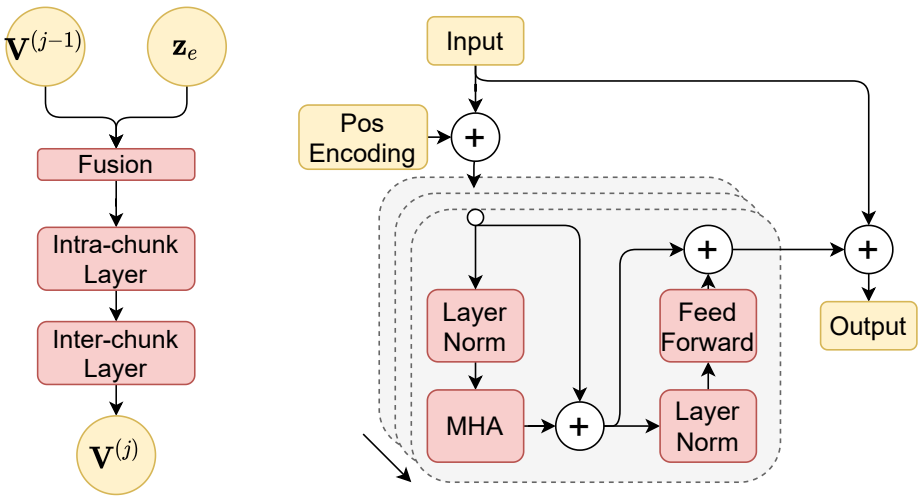

Figure 5.12: (Left) The j-th dual path transformer block. The dual-path processing module is made up of a stack of dual-path transformer blocks, where each block contains a fusion layer for processing speaker information, an intra-chunk layer for learning local dependency, and an inter-chunk layer for modeling global dependency. (Right) An intra (or inter)-chunk layer, which includes layer normalization, multi-head attention, and a feed-forward layer with residual connections. Positional encoding is applied at the beginning each intra (or inter)-chunk layer.

transformer encoder blocks [7], including layer normalization [247], multi-head attention (MHA), and a feed-forward layer with residual connections. A sinusoidal positional encoding is applied before each MHA layer. Each transformer block preserves the number of features F.

The intra-chunk layer operates on each of the S segments independently and computes the local dependency within each segment. For the j-th dual-path block, the intra-chunk layer is applied to the second dimension of the speaker-guided input tensor $\mathbf{W}^{(j)}$ by

$$\mathbf{U}_i^{(j)} = f_{\text{intra}}^{(j)}(\mathbf{W}^{(j)}[:,:,i]), i \in \{1, 2, \ldots, K\}, \tag{5.16}$$

where $f_{\text{intra}}^{(j)}$ represents the j-th intra-chunk layer, and the intra-chunk output $\mathbf{U}^{(j)}$ is obtained by concatenating $\{\mathbf{U}_i^{(j)}\}_{i=1}^K$ along the third dimension. The inter-chunk layer processes the long-term, global dependency between the segments and is applied to the third dimension of the intra-chunk layer's output

$$\mathbf{V}_i^{(j)} = f_{\text{inter}}^{(j)}(\mathbf{U}^{(j)}[:,i,:]), i \in \{1, 2, \ldots, S\}, \tag{5.17}$$

and we concatenate $\{\mathbf{V}_i^{(j)}\}_{i=1}^S$ along the second dimension to obtain the block output $\mathbf{V}^{(j)}$, which is further processed by block $j+1$. With the interleaving intra-chunk and inter-chunk layers, the model effectively models both local and global dependency in turn.

With a total of B dual-path blocks, the output $\mathbf{V}^{(B)}$ is processed by a 2D convolutional layer to increase its dimension by C folds, representing each of the C output sources with $\mathbf{M}' = \{\mathbf{M}'_c\}_{c=1}^C =$

Conv2D($\mathbf{V}^{(B)}$), where $\mathbf{M}'_c \in \mathbb{R}^{F\times K\times S}$.

Overlap-Add Finally, an overlap-add operation transforms the S segments in each $\mathbf{M}'_c$ back to T frames followed by a ReLU activation to obtain the estimated mask $\mathbf{M}_c \in \mathbb{R}^{F\times T}$ for source c.

Decoder

The decoder is a one-dimensional transposed convolutional layer that transforms the masked representation $\mathbf{H}_c = \mathbf{M}_c \odot \mathbf{H} \in \mathbb{R}^{F\times T}$ of each source c to the estimation of the time domain signal $\hat{\mathbf{s}_c} = \text{ConvTr1D}(\mathbf{H}_c) \in \mathbb{R}^L$.

5.3.3 Semi-Supervised Learning for Personalized Speech Enhancement

Prior studies on personalized speech enhancement focus on supervised training with the reference speech available. To incorporate a large unlabeled corpus, where the ground-truth target is missing, we propose a semi-supervised approach by applying a triplet speaker representation loss [248]. Given the estimated target signal $\hat{\mathbf{s}_t}$ and residual $\hat{\mathbf{s}_r}$, we compute the embeddings $\mathbf{z}_t = f_{\text{emb}}(\hat{\mathbf{s}_t})$ and $\mathbf{z}_r = f_{\text{emb}}(\hat{\mathbf{s}_r})$ with the pre-trained embedder, and compute their distances to $\mathbf{z}_e$ which is obtained from the enrollment speech. If the estimated target is separated from the residual, $\mathbf{z}_e$ should be much more similar to $\mathbf{z}_t$ than to $\mathbf{z}_r$. The triplet embedder loss is defined as

$$\mathcal{L}_{\text{emb}}(\mathbf{z}_e, \mathbf{z}_t, \mathbf{z}_r) = \max\{\|\mathbf{z}_e - \mathbf{z}_t\|_2 - \|\mathbf{z}_e - \mathbf{z}_r\|_2 + \gamma, 0\}, \tag{5.18}$$

where $\gamma = 1$ represents the margin. The embedder loss is computed along with the reconstruction loss in a supervised context in [248], while we extend its application to unlabeled data. Specifically, when the ground-truth signal is not available, we only compute the embedder loss; otherwise, we compute both the reconstruction loss $\mathcal{L}_{\text{SI-SDR}}$ and $\mathcal{L}_{\text{emb}}$. The overall objective for the semi-supervised setup is

$$\begin{aligned}\mathcal{L}_{\text{semi}} = \lambda_s \mathbb{1}[\{\mathbf{s}, \mathbf{x}, \mathbf{x}_e\} \in \mathcal{D}_l] \cdot \frac{1}{2} \sum_{c\in\{t,r\}} \mathcal{L}_{\text{SI-SDR}}(\mathbf{s}_c, \hat{\mathbf{s}_c})) \\ + \lambda_u \mathcal{L}_{\text{emb}}(\mathbf{z}_e, \mathbf{z}_t, \mathbf{z}_r),\end{aligned} \tag{5.19}$$

where λ_s, λ_u are hyperparameters controlling the two losses, $\mathbb{1}$ is the indicator function, and $\mathcal{D}_l$ represents the dataset with ground-truth target speech.

One concern with applying semi-supervised training from scratch is that it might be difficult for the model to obtain optimal separation performance quickly since the parameters are drifted away by the embedder loss in Equation (5.18). We experiment with a two-stage setup where the model is first trained with SI-SDR using only labeled data and then trained with both labeled and unlabeled data with Equation (5.19). In the supervised stage, the model learns to adapt the

speaker embedding and to extract the target speech for high-quality reconstruction. When the performance reaches the plateau, we enter the second stage by introducing the unlabeled data and the embedder loss. The goal for the second stage is to improve the model's ability to generalize with the additional unlabeled data.

5.3.4 Experimental Setup

Dataset

We train and evaluate the system with the simulated 2-speaker mixture from the *WSJ0-2mix-extr*[6] dataset. The recordings are at a sampling rate of 8 kHz. The training, development, and test set contains 20,000, 5,000, and 3,000 utterances, respectively. The mixtures are generated from speech recordings at a signal-to-noise ratio (SNR) between 0 and 5 dB. The first speaker is chosen to be the target speaker and the second speaker is treated as interference. For each target speaker, another speech recording from this speaker is randomly selected as the enrollment utterance. For the semi-supervised setup, we incorporate over 100,000 recordings from the VoxCeleb 1 corpus [249]. Since the recordings may contain background and reverberation, they are not suitable for the standard supervised setup.

Model and Training Details

First, we pre-train the speaker embedder following [231]. The embedder takes the 40-dimensional log mel-spectrogram as input. It consists of 3 BLSTM layers, each with 768 hidden units, and a linear projection head is attached to obtain an embedding with a dimensionality of 256. The network is trained on the VoxCeleb 1 & 2 [249] and the clean training partition of the Librispeech [250] corpus combined, which may contain ambiance noise or reverberation. To adapt to the memory constraint, each batch consists of 46 speakers, each speaker with 4 utterances with 4 seconds long. The network parameters are updated with the Adam optimizer [134] with an initial learning rate of 5e-4. During preliminary experiments on supervised personalized speech enhancement, we obtain the best-performing teacher network with the embedder checkpoint at 550,000 steps with an equal error rate (EER) of 6.5% on the VoxCeleb1 verification test set.

We train the extractor network with the optimal configuration from [105]. The encoder convolution has a kernel size of 16 and a stride of 8 with 256 output channels. The masking network contains two dual-path blocks. Each intra-chunk or inter-chunk module contains 8 transformer encoder layers, each layer with 8-head attention and a 1024-dimensional feed-forward projection. The decoder mirrors the encoder with the same kernel and stride sizes. We experiment with concatenation, element-wise multiplication, and addition as the embedding-mixture fusion discussed in 5.3.2.

[6] https://github.com/xuchenglin28/speaker_extraction

During training, we apply dynamic mixing [105] to generate mixtures on the fly from a random 3-second segment from each source recording. If either of the sources is less than three seconds, zero padding is applied on both ends. We also apply time-domain speed perturbing by 5% as data augmentation. We set the batch size as 1 for the experiments, and we use an Adam optimizer with an initial learning rate of 1.5e-4. The learning rate is reduced by half if the validation loss is not improved for two consecutive epochs after 85 epochs until the learning rate reaches 1e-6.

For the semi-supervised training setup, we set the weights of the two loss functions with $\lambda_s = 1$ and $\lambda_u = 0.05$. We experiment with different probabilities of data sampled from the labeled versus unlabeled corpus. For the two-stage setup, we proceed with the best-performing model from the supervised setup. We adjust the initial learning rate for the second stage to be 7.5e-5, and the aforementioned learning rate scheduler applies after 65 epochs. We encourage the readers to refer to our implementation[7].

Evaluation Details

We compare the different configurations of the proposed Exformer with previous works using the test partition of the WSJ0-2mix-extr dataset. We evaluate each system with the SI-SDR and the perceptual evaluation of speech quality (PESQ) [251] score of the model's output with respect to the reference speech.

5.3.5 Results and Discussions

Supervised Training and Fusion Configuration

Method	Fusion Type	SI-SDR (dB)	PESQ
Input	✗	2.51	1.81
TseNet [246]	Concat	14.73	3.14
SpEx+ [240]		18.2	3.49
Exformer (Ours)	Concat	19.05	3.78
	Mult	19.55	**3.82**
	Add	**19.85**	**3.82**

Table 5.7: Performance on the WSJ0-2mix-extr test set of the proposed and baseline time-domain TSE systems under supervised training, with the best performance in bold.

Table 5.7 summarizes the performance of previous methods on time-domain personalized speech enhancement as well as the proposed architecture with multiple configurations under the supervised setup. We report the mean SI-SDR and PESQ for each setup. The SI-SDR and PESQ scores for the baselines are adapted from [240].

[7] https://github.com/zhepeiw/tse_semi

All three variants of the Exformer outperform the baseline methods [240, 246]. The performance gain can be attributed to the use of the dual-path transformer blocks in the masking network which effectively model local and global dependency. Our observations are consistent with the findings of BSS systems in [105], where the performance of the transformer blocks dominates over that of the convolutional blocks.

We also study the impact of different fusion strategies for speaker embedding and mixture representation. Among the three fusion types, additive fusion has the best performance with a gain of 0.3 dB SI-SDR over element-wise multiplication and a 0.8dB gain over concatenation. Addition and multiplication share the same PESQ score slightly higher than concatenation. Adding the speaker embedding to the mixture representation is similar to applying a time-varying positional encoding in various sequential modeling tasks [7]. With the concatenation approach, a part of the dimensional space of the concatenated representation will be used for modeling the speaker embedding, hence decreasing the proportion of the vector space used for the representation for speaker extraction and may have a negative impact on the extraction quality. In contrast, addition or element-wise multiplication avoids the hard split in the vector space and leaves the optimization of the usage of the vector space for the learning process. We show in our experiments the effectiveness of adding a time-invariant speaker embedding to the latent representation, allowing the network to effectively capture the conditioning speaker information.

Semi-supervised Training and Embedder Loss

Embedder Loss	Unlabeled Percentage	Two-stage	SI-SDR (dB)	PESQ
✗	0	✗	19.85	3.82
✓		✗	19.74	**3.85**
✓		✓	20.21	3.83
✓	5	✗	19.52	3.79
		✓	20.34	3.83
✓	10	✗	19.17	3.74
		✓	**20.49**	**3.85**
✓	25	✗	19.60	3.82
		✓	20.20	3.83
✓	50	✗	17.27	3.56
		✓	19.92	3.82

Table 5.8: Personalized speech enhancement performance of the Exformer trained with the embedder loss along with multiple semi-supervised configurations.

We next study the impact of the embedder loss (5.18) and the use of unlabeled data. We report in Table 5.8 the performance of the Exformer trained on different combinations of the loss functions, the number of training stages, and the probability of unlabeled data being sampled into a training

batch. We initialize the networks in the second stage with the weights of the best-performing supervised system in Table 5.7.

Training the network from scratch using the embedder loss yields worse performance. This observation verifies our hypothesis that the embedder loss does not contribute to optimal separation performance with random initialization compared to training with reconstruction loss alone. With two-stage training, however, the extraction performance improves consistently among various percentages of applying unlabeled data. As the percentage increases, the performance reaches the peak at a probability of 10% of sampling unlabeled data in both evaluation metrics with a 0.6 dB gain of SI-SDR over the supervised baseline. The results show that the system benefits from a two-stage procedure with the embedder loss using a small ratio of unlabeled to labeled data.

5.3.6 Conclusion

We propose the Exformer, a time-domain transformer-based architecture for personalized speech enhancement. Under the supervised training setup, the Exformer significantly outperforms prior time-domain networks. More importantly, we show that the extraction performance can be further enhanced with a two-stage semi-supervised pipeline where mixtures without clean reference speech can be learned with the proposed embedder loss. By leveraging the massive amount of data without annotations, we empirically demonstrate that the semi-supervised method outperforms the supervised counterparts trained only with data samples with corresponding references. A possible future direction is to extend this approach to more complicated conditions with additional speakers, noise, and reverberation.

CHAPTER 6: CONCLUSION

6.1 SUMMARY OF CONTRIBUTIONS

In this thesis, we present data-efficient learning approaches for audio classification and separation tasks. We overcome a variety of data-related challenges in training and inference settings for audio processing. First, in Chapter 3, we present architectural designs for more efficient data usage in sound event detection. We study a scenario with training data homogeneous in terms of the number of input channels in a multi-channel speech detection task. We design a multi-view network that unrolls across both channel and temporal dimensions to overcome the lack of variability in the number of channels from the training data. The model can first be trained on a fixed number of channels and then deployed to cases with varying channel numbers, thereby reducing the effort to collect a training set covering a wide range of channels. We extend the multi-view network to an adaptive computation framework, HIDACT, for single-channel sound event detection. The model is trained to make predictions without observing information from all frequency sub-bands and to intelligently adjust the amount of computation based on the properties of the input. This technique enables processing in a challenging condition where part of the sub-band information is unavailable to the model.

Next, in Chapter 4, we investigate approaches to continually update a model to recognize new sound classes. We present a continual learning framework to efficiently train the classifier only on new data without forgetting what it has learned previously. We propose a generative replay method, where a generator is trained to produce data from the past as an economical alternative to storing previous data. Our method effectively combats catastrophic forgetting while significantly alleviating the storage burden that would otherwise exist if training along with past data. We then explore unsupervised continual learning in which few or no annotations exist. We integrate continual learning with similarity-based self-supervised algorithms to incrementally learn representations for new sound classes. This simple continual self-supervised learning framework is computationally efficient, is flexible to learn representations for an indefinite number of sound classes, and is applicable to practical use cases where only limited labels are available.

Finally, in Chapter 5, we present semi-supervised algorithms to learn sound classifiers and separators with a massive amount of data without annotations. We focus on bootstrapping-based approaches for cross-modal representation learning, where we automatically curate additional audio-text pairs from unpaired audio and text corpus using a pre-trained teacher model trained on a small set of paired data, and the curated data is used for training a student model that improves upon the teacher. We also propose this teacher-student framework for singing voice separation. In this case, the teacher model extracts vocal and accompaniment tracks from the input mixtures, which are then used as training targets for the subsequent student model. Last but not least, we consider a regularization-based semi-supervised learning scheme for personalized speech enhance-

ment, where a large scale of noisy speech mixtures without a clean reference can be learned with a speaker embedding loss. By extending the limited dataset with annotations, we empirically show the improvement of the semi-supervised algorithms upon conventional supervised baselines.

6.2 FUTURE DIRECTIONS

Our current progress can be extended further in terms of the scope of the problem and the methodology applied. Specifically, we consider the following extensions to the existing work.

6.2.1 Multi-Label Sound Classification

Our current approaches for sound classification are limited to multi-class classification where the ground truth is assumed to be one-hot. In practice, however, a single audio clip may entail multiple event classes. The scale of publicly available audio datasets that contain only a single event for each clip is limited. These datasets often contain recordings with limited duration and cover only a narrow set of sound classes. Meanwhile, research in sound recognition has recently shifted towards large-scale datasets with real-world recordings such as AudioSet [196] and FSD50K [200] in which a noticeable portion of the clips involve multiple sound classes. To fairly compare with the mainstream classification algorithms and to make our method more applicable to practical use cases, it is essential to extend our current sound classification algorithms to multi-label scenarios in order to handle challenging acoustic conditions with potentially overlapping sound events. We would like to integrate the proposed network architectures, continual, self-supervised, and semi-supervised learning frameworks into multi-label classification settings to bridge the gap toward real-world audio applications.

6.2.2 Diverse Settings and Applications for Continual Learning

For the continual learning algorithms presented in this thesis, we focus on the class-incremental setting where the model progressively learns new sound classes. As mentioned in Section 2.5, domain-incremental learning is another continual learning setting with importance. Instead of learning new sound classes over time, the goal for domain-incremental learning is to learn from a sequence of datasets, each with potentially distinct distributions. This setting is less restrictive on the dataset assumptions and is more closely related to real-world training than class-incremental learning. For instance, we may collect sound clips recorded from different microphones and acoustic conditions over time, and domain-incremental learning aims to continually learn a classifier that can recognize sound classes from recordings produced under a variety of environments that the model has seen. We would also like to further explore memory and computationally efficient algorithms (e.g., regularization-based methods, generative replay) to cope with these continual learning settings.

Besides different continual learning settings, we would also like to extend continual learning to additional audio processing tasks beyond sound event classification. We consider applications of continual learning in source separation tasks, where the model progressively learns to separate new sound classes or to separate under different acoustic conditions. Additionally, the integration of continual learning with cross-modal representation learning is also relevant. Cross-modal representation learning requires an even more substantial amount of data than learning unimodal audio representations, and it may become infeasible to train these large models with all multi-modal data at once. Continual learning approaches can help balance the trade-off between performance and efficiency when progressively updating these cross-modal representations with new data.

6.2.3 Multi-Modal Applications

In this thesis, we have presented a study on audio-text pretraining that bridges the representations of audio and natural language. As video clips with soundtracks exist at a large scale and is easily accessible [184, 196], audio-visual learning for multi-modal applications has become a central research problem. Learning representations jointly across audio, image, and natural language [191] can provide us with a more holistic perception, since information missing from one modality can be complemented by the others. It also enables cross-modal information retrieval and can improve the organization of multimedia content. Learning multi-modal representations with data-efficient approaches is a promising future direction since models involving multiple modalities require a massive amount of data to train while data aligned across different modalities may be expensive to obtain.

6.3 BROADER IMPACT

The research presented in this thesis has several contributions to the research community of artificial intelligence and machine learning. To begin with, addressing the shortage of data with high-quality annotations has been a long-lasting research problem. It requires considerable human efforts to collect, organize, and annotate the data at a large scale. To address the lack of diversity of training data, we propose algorithms that learn from a relatively homogeneous training set while generalizing well to various unseen test conditions, alleviating the need for collecting training data with comprehensive coverage. To overcome the bottleneck of the quantity of labeled training data, we propose self-supervised and semi-supervised approaches to incorporate a substantial amount of data without annotations into the learning procedure, which outperforms supervised algorithms that rely solely on annotated data. To this end, the proposed research shows the potential for training models with reduced laboring costs for human supervision of data collection. Another limitation to the development of conventional machine learning paradigms is the assumption of a fixed training dataset. This assumption may become unrealistic in practice, as new data samples can be produced at all times. We present continual learning methods that can bridge the gap between

existing learning schemes and real-world problems by incrementally incorporating knowledge from the new data while not forgetting about the past. These algorithms serve as a foundation for how industries and institutions can progressively update learned models in a lifelong fashion with minimal performance degradation.

Aside from the influence on scientific research, this thesis can also make a significant social impact from several perspectives. First, the research on sound event classification and detection is beneficial for public safety and home security systems. Events such as gunshots, glass breaking, screaming, and baby crying are easier to be recognized through audio than through visual surveillance. Accurately analyzing and detecting these events can help identify potential safety threats or enable a faster response by security and emergency services. Second, the presented research also has the potential to be applied to assistive technologies. By providing text descriptions of the recognized sound scenes and events, sound classification algorithms can make audio accessible to the community with hearing disabilities and help them understand the physical surroundings. Furthermore, our proposed research can greatly enhance the experience of remote communication. Research on source separation and speech enhancement has played an instrumental role as online meeting systems become increasingly popular since the spread of the pandemic. With the reduction of background noise and enhancement of signal quality, we enable a more accessible and frustration-free experience for online communication, improving productivity and the quality of our work and life. Last but not least, the studies of adaptive computation and continual learning are aligned with the spirit of sustainability and affordability. By reducing the computational overhead and storage requirement, the proposed solutions demonstrate the potential for deploying these algorithms on edge devices, not only reducing carbon footprint but also enabling applications under resource-constrained settings.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[4] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[5] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. [Online]. Available: https://aclanthology.org/D14-1162 pp. 1532–1543.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2015.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[8] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, jun 2017.

[9] L. JiaKai, "Mean teacher convolution system for dcase 2018 task 4," in *DCASE Challenge*, 2018.

[10] L. Delphin-Poulat and C. Plapous, "Mean teacher with data augmentation for dcase 2019 task 4 technical report," in *DCASE Challenge*, 2019.

[11] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Convolution-augmented transformer for semi-supervised sound event detection," DCASE2020 Challenge, Tech. Rep., June 2020.

[12] X. Zheng, H. Chen, and Y. Song, "Zheng ustc team's submission for dcase2021 task4 – semi-supervised sound event detection," DCASE2021 Challenge, Tech. Rep., June 2021.

[13] S. Suh, S. Park, Y. Jeong, and T. Lee, "Designing acoustic scene classification models with CNN variants," DCASE2020 Challenge, Tech. Rep., June 2020.

[14] F. Schmid, S. Masoudian, K. Koutini, and G. Widmer, "CP-JKU submission to dcase22: Distilling knowledge for low-complexity convolutional neural networks from a patchout audio transformer," DCASE2022 Challenge, Tech. Rep., June 2022.

[15] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2721–2725.

[16] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," in *Proc. Interspeech 2021*, 2021, pp. 571–575.

[17] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," 2019. [Online]. Available: https://arxiv.org/abs/1912.10211

[18] S. Oramas, F. Barbieri, O. Nieto, and X. Serra, "Multimodal deep learning for music genre classification," *Trans. Int. Soc. Music. Inf. Retr.*, vol. 1, pp. 4–21, 2018.

[19] C. Liu, L. Feng, G. Liu, H. Wang, and S. Liu, "Bottom-up broadcast neural network for music genre classification," 2019. [Online]. Available: https://arxiv.org/abs/1901.08928

[20] S. Tripathi, S. Tripathi, and H. Beigi, "Multi-modal emotion recognition on iemocap dataset using deep learning," 2018. [Online]. Available: https://arxiv.org/abs/1804.05788

[21] J. Zhao, X. Mao, and L. Chen, "Speech emotion recognition using deep 1d & 2d cnn lstm networks," *Biomed. Signal Process. Control.*, vol. 47, pp. 312–323, 2019.

[22] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming keyword spotting on mobile devices," in *Interspeech 2020*. ISCA, oct 2020. [Online]. Available: https://doi.org/10.21437%2Finterspeech.2020-1003

[23] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized Channel Attention, propagation and aggregation in TDNN based speaker verification," in *Interspeech 2020*, 2020, pp. 3830–3834.

[24] S. Pascual, A. Bonafonte, and J. Serrà, "Segan: Speech enhancement generative adversarial network," in *Interspeech*, 2017.

[25] J.-M. Valin, S. V. Tenneti, K. Helwani, U. Isik, and A. Krishnaswamy, "Low-complexity, real-time joint neural echo control and speech enhancement based on percepnet," in *ICASSP 2021*, 2021. [Online]. Available: https://www.amazon.science/publications/low-complexity-real-time-joint-neural-echo-control-and-speech-enhancement-based-on-percepnet

[26] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, pp. 1256–1266, 2018.

[27] E. Tzinis, Z. Wang, and P. Smaragdis, "Sudo rm -rf: Efficient networks for universal audio source separation," *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2020.

[28] J. Casebeer, B. Luc, and P. Smaragdis, "Multi-view networks for denoising of arbitrary numbers of channels," *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, pp. 496–500, 2018.

[29] J. Casebeer, Z. Wang, and P. Smaragdis, "Multi-view networks for multi-channel audio classification," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 940–944.

[30] Z. Wang, J. Casebeer, A. Clemmitt, E. Tzinis, and P. Smaragdis, "Sound event detection with adaptive frequency selection," *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 41–45, 2021.

[31] G. Kaissis, A. Ziller, J. Passerat-Palmbach, T. Ryffel, D. Usynin, A. Trask, I. Lima, J. V. Mancuso, F. Jungmann, M.-M. Steinborn, A. Saleh, M. R. Makowski, D. Rueckert, and R. F. Braren, "End-to-end privacy preserving deep learning on multi-institutional medical imaging," *Nat. Mach. Intell.*, vol. 3, pp. 473–484, 2021.

[32] R. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, no. 4, 1999.

[33] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, 12 2016.

[34] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2018.

[35] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: incremental classifier and representation learning," in *CVPR*, 2017.

[36] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/0efbe98067c6c73dba1250d2beaa81f9-Paper.pdf

[37] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[38] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *ECCV*, 2018.

[39] Z. Wang, Y. C. Sübakan, E. Tzinis, P. Smaragdis, and L. Charlin, "Continual learning of new sound classes using generative replay," *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 308–312, 2019.

[40] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European Conference on Computer Vision (ECCV)*, 2016.

[41] A. Dosovitskiy, J. T. Springenberg, M. A. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *NIPS*, 2014.

[42] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[43] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *ArXiv*, vol. abs/1909.11942, 2020.

[44] K. He, X. Chen, S. Xie, Y. Li, P. Doll'ar, and R. B. Girshick, "Masked autoencoders are scalable vision learners," *ArXiv*, 2021.

[45] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[46] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[47] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," in *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[48] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *ArXiv*, vol. abs/1807.03748, 2018.

[49] M. Ravanelli, J. Zhong, S. Pascual, P. Swietojanski, J. Monteiro, J. Trmal, and Y. Bengio, "Multi-task self-supervised learning for robust speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[50] Y. Gong, C.-I. J. Lai, Y.-A. Chung, and J. Glass, "Ssast: Self-supervised audio spectrogram transformer," *arXiv preprint arXiv:2110.09784*, 2021.

[51] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Self-supervised learning for general-purpose audio representation," in *International Joint Conference on Neural Networks (IJCNN)*, 2021.

[52] E. Fonseca, A. Jansen, D. P. W. Ellis, S. Wisdom, M. Tagliasacchi, J. R. Hershey, M. Plakal, S. Hershey, R. C. Moore, and X. Serra, "Self-supervised learning from automatically separated sound scenes," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2021.

[53] Z. Wang, C. Subakan, X. Jiang, J. Wu, E. Tzinis, M. Ravanelli, and P. Smaragdis, "Learning representations for new sound classes with continual self-supervised learning," *ArXiv*, vol. abs/2205.07390, 2022.

[54] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. The MIT Press, 2006. [Online]. Available: http://dblp.uni-trier.de/db/books/collections/CSZ2006.html

[55] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NIPS*, 2017.

[56] C. Baur, S. Albarqouni, and N. Navab, "Semi-supervised deep learning for fully convolutional networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2017.

[57] Q. Xie, E. H. Hovy, M.-T. Luong, and Q. V. Le, "Self-training with noisy student improves imagenet classification," *ArXiv*, vol. abs/1911.04252, 2019.

[58] P. Liang, "Semi-supervised learning for natural language," 2005.

[59] S. Grollmisch and E. Cano, "Improving semi-supervised learning for audio classification with fixmatch," *Electronics*, 2021.

[60] L. Cances and T. Pellegrini, "Comparison of deep co-training and mean-teacher approaches for semi-supervised audio tagging," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 361–365.

[61] S. Novotney and R. Schwartz, "Analysis of low-resource acoustic model self-training," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 01 2009, pp. 244–247.

[62] Z. Wang, C. Subakan, K. Subramani, J. Wu, T. F. Tavares, F. Ayres, and P. Smaragdis, "Unsupervised improvement of audio-text cross-modal representations," *ArXiv*, vol. abs/2305.01864, 2023.

[63] Z. Wang, R. Giri, U. Isik, J.-M. Valin, and A. Krishnaswamy, "Semi-supervised singing voice separation with noise self-training," in *ICASSP 2021*, 2021. [Online]. Available: https://www.amazon.science/publications/semi-supervised-singing-voice-separation-with-noise-self-training

[64] Z. Wang, R. Giri, S. Venkataramani, U. Isik, J.-M. Valin, P. Smaragdis, M. Goodwin, and A. Krishnaswamy, "Semi-supervised time domain target speaker extraction with attention," *ArXiv*, vol. abs/2206.09072, 2022.

[65] T. Heittola, A. Mesaros, and T. Virtanen, "Tau urban acoustic scenes 2022 mobile, development dataset," 2022.

[66] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*. ACM Press, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806390 pp. 1015–1018.

[67] R. Giri, S. Venkataramani, J.-M. Valin, U. Isik, and A. Krishnaswamy, "Personalized percepnet: Real-time, low-complexity target voice separation and enhancement," in *Interspeech 2021*, 2021. [Online]. Available: https://www.amazon.science/publications/personalized-percepnet-real-time-low-complexity-target-voice-separation-and-enhancement

[68] F. Su, L. Yang, T. Lu, and G. Wang, "Environmental sound classification for scene recognition using local discriminant bases and hmm," in *MM '11*, 2011.

[69] L. Vuegen, B. V. Broeck, P. Karsmakers, J. Gemmeke, B. Vanrumste, and H. V. hamme, "An mfcc-gmm approach for event detection and classification," in *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

[70] Lie Lu, Stan Z. Li, and Hong-Jiang Zhang, "Content-based audio segmentation using support vector machines," in *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, 2001, pp. 749–752.

[71] B. Uzkent, B. Barkana, and H. Çevikalp, "Non-speech environmental sound classification using svms with a new set of features," in *International Journal of Innovative Computing, Information and Control*, 2012.

[72] Jia-Ching Wang, Jhing-Fa Wang, Kuok Wai He, and Cheng-Shu Hsu, "Environmental sound classification using hybrid svm/knn classifier and mpeg-7 audio low-level descriptor," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 2006, pp. 1731–1735.

[73] S. Gunasekaran and K. Revathy, "Content-based classification and retrieval of wild animal sounds using feature selection algorithm," in *2010 Second International Conference on Machine Learning and Computing*, 2010, pp. 272–275.

[74] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[75] V. Boddapati, A. Petef, J. Rasmusson, and L. Lundberg, "Classifying environmental sounds using image recognition networks," *Procedia Computer Science*, vol. 112, pp. 2048–2056, 12 2017.

[76] S. Abdoli, P. Cardinal, and A. Lameiras Koerich, "End-to-end environmental sound classification using a 1d convolutional neural network," *Expert Systems with Applications*, vol. 136, pp. 252 – 263, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417419304403

[77] J. Kim, J. Kim, S. Lee, J. Park, and M. Hahn, "Vowel based voice activity detection with lstm recurrent neural network," in *Proceedings of the 8th International Conference on Signal Processing Systems*, 11 2016, pp. 134–137.

[78] G. Gelly and J. Gauvain, "Optimization of rnn-based speech activity detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 3, pp. 646–656, 2018.

[79] S. Adavanne, P. Pertilä, and T. Virtanen, "Sound event detection using spatial features and convolutional recurrent neural network," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 771–775.

[80] J. Sang, S. Park, and J. Lee, "Convolutional recurrent neural networks for urban sound classification using raw waveforms," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 2444–2448.

[81] H. Lim, J. Park, K. Lee, and Y. Han, "Rare sound event detection using 1d convolutional recurrent neural networks," in *Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[82] Z. Zhang, S. Xu, S. Zhang, T. Qiao, and S. Cao, "Learning attentive representations for environmental sound classification," *IEEE Access*, vol. 7, pp. 130 327–130 339, 2019.

[83] Z. Zhang, S. Xu, T. Qiao, S. Zhang, and S. Cao, "Attention based convolutional recurrent neural network for environmental sound classification," in *Pattern Recognition and Computer Vision*, Z. Lin, L. Wang, J. Yang, G. Shi, T. Tan, N. Zheng, X. Chen, and Y. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 261–271.

[84] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third 'chime' speech separation and recognition challenge: Dataset, task and baselines," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 504–511.

[85] F.-R. Stöter, A. Liutkus, and N. Ito, *The 2018 Signal Separation Evaluation Campaign*, 06 2018, pp. 293–305.

[86] I. Kavalerov, S. Wisdom, H. Erdogan, B. Patton, K. W. Wilson, J. L. Roux, and J. R. Hershey, "Universal sound separation," *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 175–179, 2019.

[87] E. Tzinis, S. Venkataramani, and P. Smaragdis, "Unsupervised deep clustering for source separation: Direct learning from mixtures using spatial information," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 81–85.

[88] P. C. Loizou, *Speech enhancement: theory and practice*. CRC press, 2013.

[89] H. Dubey, V. Gopal, R. Cutler, A. Aazami, S. Matusevych, S. Braun, S. E. Eskimez, M. Thakker, T. Yoshioka, H. Gamper, and R. Aichner, "Icassp 2022 deep noise suppression challenge," in *IEEE ICASSP*, 2022.

[90] P. Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," in *Independent Component Analysis and Blind Signal Separation: Fifth International Conference, ICA 2004, Granada, Spain, September 22-24, 2004. Proceedings 5*. Springer, 2004, pp. 494–499.

[91] M. N. Schmidt and R. K. Olsson, "Single-channel speech separation using sparse non-negative matrix factorization." in *Interspeech*, vol. 2. Citeseer, 2006, pp. 2–5.

[92] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[93] S. Choi, A. Cichocki, H.-M. Park, and S.-Y. Lee, "Blind source separation and independent component analysis: A review," *Neural Information Processing-Letters and Reviews*, vol. 6, no. 1, pp. 1–57, 2005.

[94] A. Ozerov, C. Févotte, and M. Charbit, "Factorial scaled hidden markov model for polyphonic audio representation and source separation," in *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2009, pp. 121–124.

[95] G. J. Mysore, P. Smaragdis, and B. Raj, "Non-negative hidden markov modeling of audio with application to source separation," in *Latent Variable Analysis and Signal Separation: 9th International Conference, LVA/ICA 2010, St. Malo, France, September 27-30, 2010. Proceedings 9*. Springer, 2010, pp. 140–148.

[96] S. Venkataramani, E. Tzinis, and P. Smaragdis, "A style transfer approach to source separation," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 170–174.

[97] S. Wisdom, E. Tzinis, H. Erdogan, R. Weiss, K. Wilson, and J. Hershey, "Unsupervised sound separation using mixture invariant training," in *Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[98] P. Chandna, M. Miron, J. Janer, and E. Gómez, "Monoaural audio source separation using deep convolutional neural networks," in *Latent Variable Analysis and Signal Separation: 13th International Conference, LVA/ICA 2017, Grenoble, France, February 21-23, 2017, Proceedings 13*. Springer, 2017, pp. 258–266.

[99] D. S. Williamson, Y. Wang, and D. Wang, "Complex ratio masking for monaural speech separation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 24, no. 3, pp. 483–492, 2015.

[100] J. Le Roux, G. Wichern, S. Watanabe, A. Sarroff, and J. R. Hershey, "Phasebook and friends: Leveraging discrete representations for source separation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 370–382, 2019.

[101] U. Isik, R. Giri, N. Phansalkar, J.-M. Valin, K. Helwani, and A. Krishnaswamy, "Poconet: Better speech enhancement with frequency-positional embeddings, semi-supervised conversational data, and biased loss," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2020.

[102] Z.-Q. Wang, G. Wichern, S. Watanabe, and J. Le Roux, "Stft-domain neural speech enhancement with very low algorithmic latency," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 397–410, 2023.

[103] J. R. Hershey, Z. Chen, J. L. Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 31–35, 2015.

[104] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path rnn: Efficient long sequence modeling for time-domain single-channel speech separation," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 46–50, 2020.

[105] C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi, and J. Zhong, "Attention is all you need in speech separation," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 21–25, 2021.

[106] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "Sdr – half-baked or well done?" *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[107] D. Yu, M. Kolbæk, Z. Tan, and J. H. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[108] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[109] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[110] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2014.

[111] X. Chen, H. Fan, R. B. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *ArXiv*, vol. abs/2003.04297, 2020.

[112] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[113] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *Proceedings of the International Conference on Machine Learning*, M. Meila and T. Zhang, Eds. PMLR, 2021.

[114] J.-B. Grill, F. Strub, F. Altch'e, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," *ArXiv*, vol. abs/2006.07733, 2020.

[115] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[116] Y. Ouali, C. Hudelot, and M. Tami, "An overview of deep semi-supervised learning," *ArXiv*, vol. abs/2006.05278, 2020.

[117] C. Biemann, "Unsupervised and knowledge-free natural language processing in the structure discovery paradigm," in *Ausgezeichnete Informatikdissertationen*, 2007.

[118] C. Zeno, I. Golan, E. Hoffer, and D. Soudry, "Task-agnostic continual learning using online variational bayes with fixed-point updates," *Neural Computation*, vol. 33, pp. 3139–3177, 2021.

[119] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016.

[120] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[121] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[122] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," in *ICLR*, 2019.

[123] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *NeurIPS*, 2019.

[124] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold et al., "Cnn architectures for large-scale audio classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.

[125] T. H. Vu and J.-C. Wang, "Acoustic scene and event recognition using recurrent neural networks," *Detection and Classification of Acoustic Scenes and Events*, vol. 2016, 2016.

[126] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *arXiv preprint arXiv:1606.00298*, 2016.

[127] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 3461–3466.

[128] B. Li, T. N. Sainath, R. J. Weiss, K. W. Wilson, and M. Bacchiani, "Neural network adaptive beamforming for robust multichannel speech recognition," in *Interspeech*, 2016.

[129] X. Xiao, S. Watanabe, H. Erdogan, L. Lu, J. Hershey, M. L. Seltzer, G. Chen, Y. Zhang, M. Mandel, and D. Yu, "Deep beamforming networks for multi-channel speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5745–5749.

[130] T. Inoue, P. Vinayavekhin, S. Wang, D. Wood, N. Greco, and R. Tachibana, "Domestic activities classification based on CNN using shuffling and mixing data augmentation," DCASE2018 Challenge, Tech. Rep., September 2018.

[131] R. Tanabe, T. Endo, Y. Nikaido, T. Ichige, P. Nguyen, Y. Kawaguchi, and K. Hamada, "Multichannel acoustic scene classification by blind dereverberation, blind source separation, data augmentation, and model ensembling," DCASE2018 Challenge, Tech. Rep., September 2018.

[132] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, "Darpa timit acoustic phonetic continuous speech corpus cdrom," 1993.

[133] D. Liu, P. Smaragdis, and M. Kim, "Experiments on deep learning for speech denoising," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 2685–2689, 1 2014.

[134] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[135] R. Scheibler, E. Bezzam, and I. Dokmanić, "Pyroomacoustics: A python package for audio room simulation and array processing algorithms," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 351–355.

[136] Y. Koizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "The NTT DCASE2020 challenge task 6 system: Automated audio captioning with keywords and sentence length estimation," in *DCASE Challenge*, 2020.

[137] N. Turpault, R. Serizel, A. Parag Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *DCASE Challenge*, 2019.

[138] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[139] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *NIPS*, 2015.

[140] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[141] J. Casebeer, U. Isik, S. Venkataramani, and A. Krishnaswamy, "Efficient trainable front-ends for neural speech enhancement," in *IEEE ICASSP*, 2020.

[142] G. Cerutti, R. Andri, L. Cavigelli, E. Farella, M. Magno, and L. Benini, "Sound event detection with binary neural networks on tightly power-constrained iot devices," in *ACM/IEEE ISLPED*, 2020.

[143] M. Kim and P. Smaragdis, "Bitwise neural networks for efficient single-channel source separation," in *IEEE ICASSP*, 2018.

[144] S. Kim, M. Maity, and M. Kim, "Incremental binarization on recurrent neural networks for single-channel source separation," in *IEEE ICASSP*, 2019.

[145] H.-W. Liao, J.-Y. Huang, S.-S. Lan, T.-H. Lee, Y.-W. Liu, and M.-S. Bai, "DCASE 2018 task 5 challenge technical report: Sound event classification by a deep neural network with attention and minimum variance distortionless response enhancement," in *DCASE Challenge*, 2018.

[146] J. Li, A. Mohamed, G. Zweig, and Y. Gong, "Lstm time and frequency recurrence for automatic speech recognition," in *2015 IEEE ASRU*, 2015.

[147] C.-C. Kao, M. Sun, Y. Gao, S. Vitaladevuni, and C. Wang, "Sub-Band Convolutional Neural Networks for Small-Footprint Spoken Term Classification," in *INTERSPEECH*, 2019.

[148] Q. Wang, J. Du, L.-R. Dai, and C.-H. Lee, "Joint noise and mask aware training for dnn-based speech enhancement with sub-band features," in *HSCMA*, 2017.

[149] Y. Luo, C. Han, and N. Mesgarani, "Ultra-lightweight speech separation via group communication," *arXiv preprint arXiv:2011.08397*, 2020.

[150] Y. Luo, C. Han, and N. Mesgarani, "Group communication with context codec for ultra-lightweight source separation," *arXiv preprint arXiv:2012.07291*, 2020.

[151] A. Graves, "Adaptive computation time for recurrent neural networks," *arXiv preprint arXiv:1603.08983*, 2016.

[152] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *ICPR*, 2016.

[153] M. Figurnov, M. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov, "Spatially adaptive computation time for residual networks," in *IEEE/CVF CVPR*, 2017, pp. 1790–1799.

[154] S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin, "Adaptive attention span in transformers," in *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019.

[155] J. Casebeer, J. Kaikaus, and P. Smaragdis, "Communication-cost aware microphone selection for neural speech enhancement with ad-hoc microphone arrays," *arXiv preprint arXiv:2011.07348*, 2020.

[156] C. Eyzaguirre and Á. Soto, "Differentiable adaptive computation time for visual reasoning," in *IEEE/CVF CVPR*, 2020.

[157] N. Whitehead and A. Fit-Florea, "Precision & performance: floating point and ieee 754 compliance for nvidia gpus," *rn (A+B)*, 2011.

[158] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge," *IEEE/ACM TASLP*, 2018.

[159] J. C. Schlimmer and D. Fisher, "A case study of incremental concept induction," in *AAAI*, vol. 86, 1986, pp. 496–501.

[160] R. S. Sutton, S. D. Whitehead et al., "Online learning with random representations," in *Proceedings of the Tenth International Conference on Machine Learning*, 1993, pp. 314–321.

[161] M. B. Ring, "Child: A first step towards continual learning," *Machine Learning*, vol. 28, no. 1, pp. 77–104, 1997.

[162] Y. C. Sübakan, O. Koyejo, and P. Smaragdis, "Learning the base distribution in implicit generative models," *CoRR*, vol. abs/1803.04357, 2018. [Online]. Available: http://arxiv.org/abs/1803.04357

[163] E. Tzinis, S. Venkataramani, Z. Wang, Y. C. Sübakan, and P. Smaragdis, "Two-step sound source separation: Training on learned latent targets," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 31–35, 2020.

[164] Y. Hsu, Y. Liu, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," *CoRR*, vol. abs/1810.12488, 2018. [Online]. Available: http://arxiv.org/abs/1810.12488

[165] M. Lucic, K. Kurach, M. Michalski, O. Bousquet, and S. Gelly, "Are gans created equal? a large-scale study," in *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. USA: Curran Associates Inc., 2018. [Online]. Available: http://dl.acm.org/citation.cfm?id=3326943.3327008 pp. 698–707.

[166] M. D.Hoffman and M. J. Johnson, "ELBO surgery: yet another way to carve up the variational evidence lower bound," in *NIPS workshop for approximate Bayesian inference*, Dec. 2016.

[167] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *CoRR*, vol. abs/1611.02648, 2016. [Online]. Available: http://arxiv.org/abs/1611.02648

[168] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: an unsupervised and generative approach to clustering," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 1965–1972.

[169] J. Tomczak and M. Welling, "Vae with a vampprior," in *International Conference on Artificial Intelligence and Statistics*, 2018, pp. 1214–1223.

[170] A. Rios and L. Itti, "Closed-loop GAN for continual learning," *CoRR*, vol. abs/1811.01146, 2018. [Online]. Available: http://arxiv.org/abs/1811.01146

[171] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *European Conference on Computer Vision (ECCV)*, 2018.

[172] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations (ICLR)*, 2019.

[173] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[174] P.-H. Chi, P.-H. Chung, T.-H. Wu, C.-C. Hsieh, S.-W. Li, and H. yi Lee, "Audio albert: A lite bert for self-supervised learning of audio representation," *IEEE Spoken Language Technology Workshop (SLT)*, 2021.

[175] E. Fini, V. G. T. da Costa, X. Alameda-Pineda, E. Ricci, K. Alahari, and J. Mairal, "Self-supervised models are continual learners," *arXiv preprint arXiv:2112.04215*, 2021.

[176] D. Madaan, J. Yoon, Y. Li, Y. Liu, and S. J. Hwang, "Representational continuity for unsupervised continual learning," in *International Conference on Learning Representations*, 2022.

[177] Z. Ni, S. Tang, and Y. Zhuang, "Self-supervised class incremental learning," *arXiv preprint arXiv:2111.11208*, 2021.

[178] S. Zhang, G. Shen, and Z. Deng, "Self-supervised learning aided class-incremental lifelong learning," *ArXiv*, 2020.

[179] S. Kessler, B. Thomas, and S. Karout, "An adapter based pre-training for efficient and scalable self-supervised speech representation learning," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2022.

[180] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," in *Advances in Neural Information Processing Systems*, 2020.

[181] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple augmentation method for automatic speech recognition," in *INTERSPEECH*, 2019.

[182] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *ACM International Conference on Multimedia*, 2014.

[183] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2018.

[184] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "Vggsound: A large-scale audio-visual dataset," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[185] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "Speechbrain: A general-purpose speech toolkit," *ArXiv*, 2021.

[186] J. Smith, J. Tian, Y.-C. Hsu, and Z. Kira, "A closer look at rehearsal-free continual learning," *ArXiv*, 2022.

[187] J. Gui, T. Chen, Q. Cao, Z. Sun, H. Luo, and D. Tao, "A survey of self-supervised learning from multiple perspectives: Algorithms, theory, applications and future trends," 2023.

[188] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, 2021, pp. 8748–8763.

[189] L. Yuan, D. Chen, Y.-L. Chen, N. C. F. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, C. Liu, M. Liu, Z. Liu, Y. Lu, Y. Shi, L. Wang, J. Wang, B. Xiao, Z. Xiao, J. Yang, M. Zeng, L. Zhou, and P. Zhang, "Florence: A new foundation model for computer vision," *ArXiv*, vol. abs/2111.11432, 2021.

[190] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," 2021.

[191] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "Audioclip: Extending clip to image, text and audio," 2021.

[192] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang, "Clap: Learning audio concepts from natural language supervision," *arXiv preprint arXiv:2206.04769*, 2022.

[193] H.-H. Wu, P. Seetharaman, K. Kumar, and J. P. Bello, "Wav2clip: Learning robust audio representations from clip," 2022.

[194] Y. Zhao, J. Hessel, Y. Yu, X. Lu, R. Zellers, and Y. Choi, "Connecting the dots between audio and text without parallel data through visual knowledge transfer," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 4492–4507.

[195] Y. Wu*, K. Chen*, T. Zhang*, Y. Hui*, T. Berg-Kirkpatrick, and S. Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2023.

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

[197] A. Andonian, S. Chen, and R. Hamid, "Robust cross-modal representation learning with progressive self-distillation," in *CVPR 2022*, 2022.

[198] Y. Gao, J. Liu, Z.-H. Xu, T. Wu, W. Liu, J. jin Yang, K. Li, and X. Sun, "Softclip: Softer cross-modal alignment makes clip stronger," *ArXiv*, vol. abs/2303.17561, 2023.

[199] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp. 85–92.

[200] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "Fsd50k: An open dataset of human-labeled sound events," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 829–852, 2022.

[201] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: an audio captioning dataset," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 736–740, 2019.

[202] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019. [Online]. Available: https://aclanthology.org/N19-1011 pp. 119–132.

[203] I. Martín-Morató and A. Mesaros, "What is the ground truth? reliability of multi-annotator data for audio tagging," *2021 29th European Signal Processing Conference (EUSIPCO)*, pp. 76–80, 2021.

[204] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6 pp. 38–45.

[205] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," *CoRR*, vol. abs/1706.09588, 2017. [Online]. Available: http://arxiv.org/abs/1706.09588

[206] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018, pp. 106–110.

[207] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix - a reference implementation for music source separation," *Journal of Open Source Software*, 2019. [Online]. Available: https://doi.org/10.21105/joss.01667

[208] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," in *ISMIR*, 2018.

[209] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020, deezer Research. [Online]. Available: https://doi.org/10.21105/joss.02154

[210] L. Prétet, R. Hennequin, J. Royo-Letelier, and A. Vaglio, "Singing voice separation: A study on training data," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 506–510, 2019.

[211] V. S. Kadandale, J. F. Montesinos, G. Haro, and E. Gómez, "Multi-channel u-net for music source separation," 2020.

[212] Y. Liu, B. Thoshkahna, A. A. Milani, and T. Kristjansson, "Voice and accompaniment separation in music using self-attention convolutional neural network," *ArXiv*, vol. abs/2003.08954, 2020.

[213] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Demucs: Deep extractor for music sources with extra unlabeled data remixed," *ArXiv*, vol. abs/1909.01174, 2019.

[214] A. Défossez, N. Usunier, L. Bottou, and F. R. Bach, "Music source separation in the waveform domain," *ArXiv*, vol. abs/1911.13254, 2019.

[215] E. Nachmani, Y. Adi, and L. Wolf, "Voice separation with an unknown number of multiple speakers," *ArXiv*, vol. abs/2003.01531, 2020.

[216] D. Samuel, A. Ganeshan, and J. Naradowsky, "Meta-learning extractors for music source separation," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 816–820, 2020.

[217] C. Hsu and J. R. Jang, "On the improvement of singing voice separation for monaural recordings using the mir-1k dataset," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 2, pp. 310–319, 2010.

[218] A. Liutkus, D. Fitzgerald, Z. Rafii, B. Pardo, and L. Daudet, "Kernel additive models for source separation," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4298–4310, 2014.

[219] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[220] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 261–265.

[221] A. Cohen-Hadria, A. Röbel, and G. Peeters, "Improving singing voice separation using deep u-net and wave-u-net with data augmentation," *2019 27th European Signal Processing Conference (EUSIPCO)*, pp. 1–5, 2019.

[222] D. Stoller, S. Ewert, and S. Dixon, "Adversarial semi-supervised audio source separation applied to singing voice extraction," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2391–2395.

[223] M. Michelashvili, S. Benaim, and L. Wolf, "Semi-supervised monaural singing voice separation with a masking network trained on synthetic mixtures," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 291–295, 2019.

[224] S. I. Mimilakis, K. Drossos, and G. Schuller, "Unsupervised interpretable representation learning for singing voice separation," *ArXiv*, vol. abs/2003.01567, 2020.

[225] S. I. Mimilakis, K. Drossos, and G. Schuller, "Revisiting representation learning for singing voice separation with sinkhorn distances," *ArXiv*, vol. abs/2007.02780, 2020.

[226] P. Seetharaman, G. Wichern, J. L. Roux, and B. Pardo, "Bootstrapping deep music separation from primitive auditory grouping principles," *ArXiv*, vol. abs/1910.11133, 2019.

[227] D. S. Park, Y. Zhang, Y. Jia, W. Han, C.-C. Chiu, B. Li, Y. Wu, and Q. V. Le, "Improved noisy student training for automatic speech recognition," *ArXiv*, vol. abs/2005.09629, 2020.

[228] F. Pishdadian, G. Wichern, and J. L. Roux, "Finding strength in weakness: Learning to separate sounds with weak supervision," *ArXiv*, vol. abs/1911.02182, 2019.

[229] I. Smule, "DAMP-VSEP: Smule Digital Archive of Mobile Performances - Vocal Separation," *Zenodo*, Oct. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3520973

[230] E. Manilow, P. Seetharman, and J. Salamon, *Open Source Tools & Data for Music Source Separation*. https://source-separation.github.io/tutorial, Oct. 2020. [Online]. Available: https://source-separation.github.io/tutorial

[231] Q. Wang, H. Muckenhirn, K. Wilson, P. Sridhar, Z. Wu, J. R. Hershey, R. A. Saurous, R. J. Weiss, Y. Jia, and I. L. Moreno, "VoiceFilter: Targeted Voice Separation by Speaker-Conditioned Spectrogram Masking," in *Interspeech*, 2019.

[232] K. Žmolíková, M. Delcroix, K. Kinoshita, T. Ochiai, T. Nakatani, L. Burget, and J. Černocký, "Speakerbeam: Speaker aware neural network for target speaker extraction in speech mixtures," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 4, 2019.

[233] Q. Wang, I. L. Moreno, M. Saglam, K. Wilson, A. Chiao, R. Liu, Y. He, W. Li, J. Pelecanos, M. Nika, and A. Gruenstein, "VoiceFilter-Lite: Streaming Targeted Voice Separation for On-Device Speech Recognition," in *Interspeech*, 2020.

[234] C. Xu, W. Rao, C. E. Siong, and H. Li, "Optimization of speaker extraction neural network with magnitude and temporal spectrum approximation loss," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[235] X. Ji, M. Yu, C. Zhang, D. Su, T. Yu, X. Liu, and D. Yu, "Speaker-aware target speaker enhancement by jointly learning with speaker embedding extraction," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[236] M. Delcroix, T. Ochiai, K. Žmolíková, K. Kinoshita, N. Tawara, T. Nakatani, and S. Araki, "Improving speaker discrimination of target speech extraction with time-domain speakerbeam," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[237] S. He, H. Li, and X. Zhang, "Speakerfilter: Deep learning-based target speaker extraction using anchor speech," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[238] J. Zhao, S. Gao, and T. Shinozaki, "Time-Domain Target-Speaker Speech Separation with Waveform-Based Speaker Embedding," in *Interspeech*, 2020.

[239] C. Xu, W. Rao, E. S. Chng, and H. Li, "Spex: Multi-scale time domain speaker extraction network," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 28, 2020.

[240] M. Ge, C. Xu, L. Wang, C. E. Siong, J. Dang, and H. Li, "SpEx+: A Complete Time Domain Speaker Extraction Network," in *Interspeech*, 2020.

[241] M. Ge, C. Xu, L. Wang, C. E. Siong, J. Dang, and H. Li, "Multi-stage speaker extraction with utterance and frame-level reference signals," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[242] S. Yuan, Z. Wang, U. Isik, R. Giri, J.-M. Valin, M. Goodwin, and A. Krishnaswamy, "Improved singing voice separation with chromagram-based pitch-aware remixing," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[243] E. Tzinis, Y. Adi, V. K. Ithapu, B. Xu, and A. Kumar, "Continual self-training with bootstrapped remixing for speech enhancement," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[244] A. Sivaraman, S. Kim, and M. Kim, "Personalized speech enhancement through self-supervised data augmentation and purification," in *Interspeech*, 2021.

[245] L. Wan, Q. Wang, A. Papir, and I. Lopez-Moreno, "Generalized end-to-end loss for speaker verification," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[246] C. Xu, W. Rao, C. E. Siong, and H. Li, "Time-domain speaker extraction network," *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019.

[247] J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv*, vol. abs/1607.06450, 2016.

[248] S. Mun, S. Choe, J. Huh, and J. S. Chung, "The sound of my voice: Speaker representation loss for target voice separation," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

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

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

[251] A. Rix, J. Beerends, M. Hollier, and A. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, 2001.