Interpreting Intermediate Feature Representations of Raw-Waveform Deep CNNs by Sonification

Sarthak Yadav

Submitted in fulfilment of the requirements for the
Degree of Master of Science by Research

School of Computing Science
College of Science and Engineering
University of Glasgow

November 2021

Abstract

The majority of the recent works that address the interpretability of raw waveform based deep neural networks (DNNs) for audio processing focus on interpreting spectral and frequency response information, often limiting to visual and signal theoretic means of interpretation, solely for the first layer. This work proposes *sonification*, a method to interpret intermediate feature representations of sound event recognition (SER) 1D-convolutional neural networks (1D-CNNs) trained on raw waveforms by mapping these representations back into the discrete-time input signal domain, highlighting substructures in the input that maximally activate a feature map as intelligible acoustic events. Sonification is used to compare supervised and contrastive self-supervised feature representations, observing how the latter learn more acoustically discernible representations, especially in the deeper layers. A metric to quantify acoustic similarity between the interpretations and their corresponding inputs is proposed, and a layer-by-layer analysis of the trained feature representations using this metric supports the observations made.

Contents

List of Tables

List of Figures

Acknowledgements

I want to thank my supervisor, Dr. Mary Ellen Foster, for their expertise, advice, words of encouragement, and patience throughout the process. I would also like to thank Dr. Alessandro Vinciarelli for a discussion that proved to be a significant course correction in this study and my friend, Perla Sai Raj Kishore, for their valuable time and discussions. Finally, I want to thank my friends at SAWB 502 for making things less isolating, especially in the early days.

This research was supported by the *TPU Research Cloud (TRC) program*, a Google Research initiative, for generously providing access to TPU v2 and v3 devices. Without their support, experimenting with the large-scale datasets used in this work would not have possible given the time constraints.

Declaration

With the exception of chapters 1, 2 and 3, which contain introductory material, all work in this thesis was carried out by the author unless otherwise explicitly stated. The content of this thesis, especially the figures, was also utilized in a research article that is under review at the time of writing (Yadav & Foster, 2022).

Chapter 1

Introduction

Recently, interpretability of deep neural networks (DNNs) has garnered a lot of attention. DNNs are capable of learning very expressive feature hierarchies and the recent trend of moving away from the divide-and-conquer approach where different modules are optimized separately to deeper networks with end-to-end training further increases abstraction and trades interpretability for better performance; a justified pursuit, but not without detriment. This is only further aggravated by the recent rise of methods of self-supervision for learning representations from noisy/unlabelled data with insufficient exploration of the aspects in which these representations differ from traditional supervised ones. As a result, the need for interpretability of DNNs continues to rise.

Visualizations are the primary means of interpreting DNNs and the feature hierarchies they learn, irrespective of data modality or DNN architecture design, and they have been utilized for visual interpretation of attention mechanisms (Woo et al., 2018; Vaswani et al., 2017; Vig, 2019; Vig & Belinkov, 2019) as well as intermediate feature representations of DNNs (Erhan et al., 2009; Simonyan et al., 2013; Zeiler & Fergus, 2014; Springenberg et al., 2015; Mahendran & Vedaldi, 2015; Smilkov et al., 2017; Selvaraju et al., 2020). And rightfully so; the visual cortex is our most essential and complex sensory perception system (Kandel et al., 2000; Sternberg et al., 2012), possibly making visualizations more immediate than other forms of interpretation.

The domain of audio and speech processing also relies extensively on visualizations to interpret and understand how DNNs work. A large body of work attempts to interpret representations learned by DNNs filters and parameters through visual means (Palaz et al., 2013, 2015; Golik et al., 2015; Verma & Schafer, 2016; Krug & Stober, 2018; Muckenhirn et al., 2018c; Palaz et al., 2019), whether directly or indirectly. However, one could argue that if the pertinent question is ascertaining what these DNNs learn and model, solely relying on visual means of interpretation might not be optimal. Visualizations work well for the computer vision domain, where most of the work on interpretability is carried out, possibly since interpretations share the same modality as the input space and allow us to observe and connect elements from the interpretation to the input signal more seamlessly. In the same way, wouldn't the audio and speech processing

domain benefit from interpreting intermediate feature representations in the audio input space where they can be directly perceived as intelligible acoustic elements over *solely* relying on visual means of description? Existing work that attempt to address this issue are few and far between (Choi et al., 2015, 2016; Li et al., 2020; Begus & Zhou, 2021).

This is the primary motivation behind the proposed approach: to interpret intermediate feature representations of deep neural networks in the audio input space.

More specifically, the proposed work:

1. Revisits transposed convolutions, also known as Deconvolutions[1] in the deep learning literature, and formulates *sonification*, which is the process of interpreting intermediate feature representations of raw waveform based 1D convolutional neural networks (CNNs) (LeCun et al., 1998) for sound event recognition (SER) by mapping them back into the discrete-time audio input space, such that they can be listened to as well as visualized. (Chapter 5).

2. Evaluates deep representations learned by a contrastive self-supervised learning process and compare them with those learned by supervised learning using sonification (Chapter 5).

3. Quantifies how interpretations compare with corresponding input signals over the depth of a CNN, establishing key differences between contrastive self-supervised and supervised representations (Chapter 6).

To this end, a 1D-CNN architecture which directly ingests raw waveform signals independent of specific feature front-ends is proposed and thoroughly evaluated on several benchmark datasets, with the objective of establishing a sufficiently performant baseline (Chapter 4). From this point, when CNNs are mentioned, the author refers to 1D-CNNs, unless explicitly stated otherwise.

The reader might have a natural question: why work with raw waveform signals directly and go through the process of evaluating a new neural architecture? The reasoning behind this design decision is as follows.

The outstanding performance of CNNs as demonstrated by Krizhevsky et al. (2012) on the ImageNet benchmark (Deng et al., 2009) was a key contributor in the resurgence of deep learning. As opposed to previous approaches which utilized handcrafted features (such as SIFT (Lowe, 1999)), CNNs directly modelled raw images with minimal preprocessing. The audio and speech processing domain has long relied upon hand crafted features, such as the Mel and the CQT filterbanks (Brown, 1991; Brown & Puckette, 1992), and has recently seen resurgence of several methods that attempt to directly processing raw waveform signal by integrating the feature extraction process in the DNN itself (Palaz et al., 2013, 2015; Golik et al., 2015; Verma

[1]Not to be confused with the deconvolution operation in signal processing domain.

& Schafer, 2016; Collobert et al., 2016; Papyan et al., 2017; Kabil et al., 2018; Muckenhirn et al., 2018b,c,a; Ravanelli & Bengio, 2018b,a; Ravanelli & Bengio, 2019; Palaz et al., 2019; Muckenhirn et al., 2019; Loweimi et al., 2019; Chowdhury & Ross, 2020; Zeghidour et al., 2021), hence making interpretability of raw waveform modelling DNNs of keen interest.

An additional reason is that, as opposed to modelling input in the frequency domain, modelling raw waveform signals directly also allows us to bypass the need for further processing when inverting representations into the audio input space, along with preserving phase information which is often discarded by magnitude-only spectral features.

The structure of the thesis is as follows: next is an in-depth discussion of relevant existing works in Chapter 2, followed by a brief discussion of all prerequisite concepts needed in Chapter 3. To evaluate *sonification*, an easy-to-follow baseline 1D-CNN feature encoder trained using supervised and self-supervised contrastive learning paradigm is needed, this is covered in-depth in Chapter 4. Finally, Chapter 5 proposes the *sonification* process and lists the observations made from evaluating generated interpretations, and Chapter 6 covers supporting empirical experiments as well as proposes a method of quantifying similarity between the interpretation and the corresponding input signal.

Motivation

The author's experience as a research engineer at an AI startup working on large-scale speaker recognition and verification systems for Indian law enforcement agencies led to their interest in the interpretability of audio processing DNNs. Having led a team of engineers on several mission-critical, large-scale commercial and government projects in the audio and computer vision domain and building interpretable systems to better support the decisions made by such DNNs, the author is confident in making the following statements:

1. Interpretability of failure modes significantly improves the confidence of stakeholders in the system, especially for sensitive and critical applications. A large majority of the such clients will prefer a worse performing albeit interpretable system over one that performs slightly better.

2. AI systems always surprise you in production: interpretability and explainability of such systems is essential to bolster your confidence in the system.

3. When it comes to debugging large DNNs, every bit counts, and interpretable systems make for happy ML developers.

Chapter 2

Related Works

This chapter covers relevant related work done on interpretability of deep neural networks (DNNs) from the perspective of audio and speech processing. Related works are divided on the basis of their core approach for interpretation. For every approach, a broad overview is provided along with a more detailed assessment of select publications.

2.1 By visualizing filters and their frequency responses

In the domain of audio and speech processing using DNNs, several works visualize filters learned from spectrogram inputs (Huang et al., 2015; Krug & Stober, 2018, 2019). Huang et al. (2015) visualize spectro-temporal filters (which are essentially filters in the frequency-time domain, similar to a spectrogram) while Krug & Stober (2019) learn topographic filter maps in the first convolutional layer by encouraging filters to learn similar concepts in a square neighbourhood within predefined grids, constraining the optimization to the sum of weighted reciprocal euclidean distance similarities within a neighbourhood.

Most of the work on interpretability is done on raw waveform based ASR models, visualizing filter frequency responses for the first convolutional layer (Palaz et al., 2013, 2015; Golik et al., 2015; Verma & Schafer, 2016). Palaz et al. (2013, 2015) analyze frequency responses of the first convolutional layer and show that it learns matching filters. Palaz et al. (2015) analyze the mean frequency response for a given vowel (denoted f_{vowel}). To this end, they sum the frequency responses of the five most firing filters weighted by the number of times the filter was triggered normalized by total number of appearances:

$$f_{vowel} = \sum_i \frac{n_i}{\sum_j n_j} F_i, \tag{2.1}$$

where F_i is the frequency response of filter i, n_i is the number of times filter i is triggered normalized by total appearances $\sum_j n_j$.

By visualizing the magnitude spectrum of the filters sorted by the estimated center frequency,

Golik et al. (2015) were able to interpret the first as well as the second convolutional layer. They showed that the majority of the filters of the first convolutional layer have learned narrow bandpass filter, performing a time-frequency decomposition similar to FIR filter banks, while the second convolutional layer detected onset/offset of a phonetic event captured by dynamics of formant information, with some filters acting similar to MRASTA filtering (Hermansky & Fousek, 2005).

More recently, Palaz et al. (2019) show that filters in the first convolutional layer model formant related information to learn a phone-discriminant spectral dictionary from sub-segmental speech, a notion that is also supported by Mallat (2016); Papyan et al. (2017); Kabil et al. (2018); Muckenhirn et al. (2018b). They proposed an end-to-end acoustic model that learns to estimate HMM states class conditional probabilities using a CNN that accepts raw waveform signal as input. They also proposed using cumulative frequency response of all the filters in the first layer to demonstrate how their proposed method does not learn constant Q bandpass filter banks, unlike fixed auditory filterbanks like Mel or Bark scale filters.

Muckenhirn et al. (2018c,b) visualize and interpret the cumulative frequency response of the first convolutional layer with n filters (Equation 2.2) trained on raw waveforms, and show how fundamental frequency and formant information is modelled by the filters, also demonstrating how the CNN model is modelling information over 2000 Hz.

$$F_{cum} = \sum_{k=1}^{n} \frac{F_k}{||F_k||}, \tag{2.2}$$

where F_k represents the frequency response of an individual filter.

2.2 By using gradient-based relevance signals

Several works propose gradient-based interpretation of intermediate feature maps of raw waveform CNNs.

Krug & Stober (2018) examine activation maximization (Erhan et al., 2009) with ℓ_1 and ℓ_2 regularization as well as sensitivity analysis (Gevrey et al., 2003) for local inspection on a fully-convolutional automatic speech recognition model based on the Wav2Letter (Collobert et al., 2016) architecture trained on spectrogram inputs. Spectrogram frames of specific width and containing characters *a-z* were analysed. They proposed a novel approach for global introspection, named normalized averaging of aligned inputs (NAvAI). The NAvAI approach averages all spectrogram frames predicting the same character before introspection, thereby requiring proper alignment of the frames with the predicted characters, which is a part of their proposed approach. This is followed by subtracting the mean of spectrogram frames belonging to the same character, and finally local and global introspection.

A popular method in the computer vision domain to interpret intermediate layers of CNNs

is guided backpropagation (Springenberg et al., 2015), which is the basis of the relevance signals approach proposed by (Muckenhirn et al., 2018a, 2019; Chowdhury & Ross, 2020) for raw waveform CNNs. In particular, Muckenhirn et al. (2018a, 2019) proposed guided backpropagation based relevance signals to inspect and visualize the impact of each frequency bin on the output of a 1D-CNN classifier trained for phone classification and speaker identification on the TIMIT dataset (Garofolo, 1993). Their approach computes gradient signal of output unit of a desired class with respect to each frequency bin in a given input raw waveform signal's frequency domain representation obtained using a fourier transform. The relevance map itself is obtained by taking inverse of the short term fourier transform of the gradient signal obtained earlier. Case studies conducted on speaker identification and phone classification tasks revealed that the relevance signal, while not interpretable by the human ear, contains periodicity information related to the input signal, and spectral dictionary based analysis on the relevance signal showed that the CNN models formant (on the phone recognition task) and F0 (on the speaker identification task) information.

2.3 By using specialized audio frontends

The advent of custom front end models for ingesting raw waveform signals without using predefined features also fosters improved interpretability. SincNet features were originally proposed for speaker recognition (Ravanelli & Bengio, 2018b) and later adapted for automatic speech recognition (ASR) (Ravanelli & Bengio, 2019). As opposed to general convolution kernel, SincNet kernels are parameteric in nature, and only consist of two parameters, the low and high cut-off frequencies, irrespective of filter length. Fewer parameters aid the interpretability of the first SincNet layer that directly models the raw waveform signal, and was analyzed in depth in (Ravanelli & Bengio, 2018a).

Inspired by SincNet features, Loweimi et al. (2019) proposed modulated kernel-based filters, assessing the performance and improved interpretability of the proposed triangular (square of the *sinc* function), gammatone and gaussian kernel based filters that directly model raw waveform signal. By visualizing the distribution of filter center frequencies, Loweimi et al. (2019) demonstrated how more filters operated in frequency regions below 2000 Hz, which is consistent with how perceptual scales (such as the Mel scale) inspired by the human auditory system work.

Recently, Chowdhury & Ross (2020) proposed DeepVox, a 1D-CNN based front-end for learning noise robust speech filterbanks directly from raw waveforms, without imposing and constraints on the design of the filters. DeepVox utilizes 1D dilated convolutions that learn features only from within a speech frame, and not across multiple frames. They estimate relevance signals for each feature bank using guided backpropagation, and use pitch-contour analysis to demonstrate that their proposed feature banks capture F0 contours of the input signal for most

of the phonemes. Power Spectral Density estimation shows that DeepVOX features model frequency range between 0 to 4000 Hz, while offering better modelling performance in the 2000 to 4000 Hz range.

2.4 By inverting representations into the audio input space to determine what the DNNs hear

Similar to the proposed work, DeconvNets were used by Choi et al. (2015, 2016) to interpret intermediate representations of 2D-CNNs trained on spectrogram inputs for music classification, a process they termed "auralisation". This approach allowed them to treat the input space as a pixel representation in the frequency domain, facilitating analysis. Choi et al. (2016) train a 5-layer 2D-CNN with interleaving max pool layers trained on a music genre classification dataset taken from 8000 songs spanning three genres (ballad, dance and hiphop). Using deconvolutions, they demonstrated concepts learned by feature maps, such as features that model onset of percussive instruments, extraction of harmonic components, with activations becoming sparser with depth. Using a correlation study they demonstrate how key, chord and instrument variations affect feature activity. Initial layers show more selectivity towards variations, whereas deep layers demonstrate similar average correlation for all the variations, indicating the contribution of high-level features.

More recently, Li et al. (2020) proposed interpreting intermediate representations learned by automatic speech recognition models in the discrete-time input space by using a reconstruction model as a probe. More specifically, they proposed using a Highway network (Srivastava et al., 2015; Zilly et al., 2017) with 4 hidden layers that is trained to reconstruct the input log mel-filterbanks from intermediate feature maps of the layer being probed by minimizing ℓ_1 loss. Linear spectrograms were obtained from these reconstructed mel-filterbanks using pseudo-inverse, followed by Griffin Lim (Griffin & Lim, 1984; Perraudin et al., 2013) based phase estimation to finally obtain audio waveforms. Li et al. (2020) evaluated how speaker information modelling varies with model depth. At the cost of being compatible with most DNN architectures, their approach requires training a separate reconstruction network for each layer in a DNN, and would hence scale extremely poorly with very deep neural networks.

Begus & Zhou (2021) proposed interpreting intermediate representations using Generative Adversarial Networks (Goodfellow et al., 2020), interpreting a WaveGAN (Donahue et al., 2019) model trained on raw waveform data in the time-domain by using a separate Categorical InfoWaveGAN (ciwGAN) model (Beguš, 2021) as a probe. Simple speech recognition tasks of detecting the presence of [s] and reduplication are used for training. Choosing GAN framework allows them to manipulate and interpolate latent variables to marginal values outside the training range and create a high occurence of a desired linguistic unit, allowing them to observe how these changes propagate throughout the convolutional layers in the generator, specifically

for the ciwGAN architecture, which can accept categorical code input c in addition to the latent variable z. This facilitates extraction of duration, intensity, fundamental frequency (F0) and formant information, and correlating them with the final output to check which layers of the Generator network model these properties. Averaged ReLU activations from a feature map yield interpretable time-series data in the GAN framework, and is also demonstrated.

2.5 Outlook

It is evident that majority of the existing work in the audio processing domain only interprets and visualizes spectral information modeled by the first convolution layer, with works that focus on intermediate representations in the time-domain few and far between. The work summarised above finds crucial analytical and signal theoretic insights into the workings of CNNs trained on audio and speech data (such as filter frequency responses, formation of phone-discriminant spectral dictionaries), but the interpretations do not bridge representations with acoustic elements in the input in most cases. This is unlike computer vision, where interpretations share the same modality as the input space and allow us to observe and connect elements more seamlessly. The proposed work utilizes transposed convolutions, a.k.a. deconvolutions (Zeiler et al., 2011) to map intermediate feature maps back into the discrete-time signal input space, highlighting maximally-activating substructures in the input space corresponding to the feature of interest in the form of intelligible acoustic events. Given that this work falls into the same category as 2.4, and the concept of deconvolutions has already been applied once for interpreting acoustic models, we would like to highlight the following core differences between the proposed approach and previous works:

No reconstruction model required

This work does not utilize a separate reconstruction model for inverting intermediate feature representations into the audio space, as opposed to Li et al. (2020); Begus & Zhou (2021), and as a result does not require any additional training to provide interpretations.

Direct conversion into the audio waveform

This work directly ingests raw waveform signals, and thus inverting representations directly takes them into the discrete-time audio signal space, unlike the work of Li et al. (2020) which requires a substantial number of steps and Choi et al. (2015, 2016), where the inversion goes back to the frequency domain. The proposed work, since it models raw waveform signals directly, can possibly exploit phase information in the signal, which the above works cannot as spectrograms discard phase spectrum.

Comparing self-supervised and supervised representations

This work evaluates and compares both self-supervised and supervised representations, unlike previous approaches, which only take a single training paradigm in consideration.

Quantifying similarity between interpretations and their corresponding inputs

This work proposes a measurement of similarity between the interpretations and their corresponding inputs based on spectral analysis of both the signals. This is not covered by previous works.

Chapter 3

Prerequisites

This chapter aims to give the reader a refresher of the concepts used in upcoming chapters of this work, focusing on the breadth and pointing the reader in the right direction if more details are warranted. Since this work focuses on interpreting convolutional neural networks (1D-CNNs), we begin with a primer on convolutions and max-pooling operations, as well as DeconvNets (Zeiler et al., 2011), with some information on implementation details in modern deep learning frameworks as well as a light review of CNNs in the audio processing literature. This is followed by a brief overview of Sound Event Recognition and the two core datasets utilized in this work: AudioSet (Gemmeke et al., 2017) and FSD50K (Fonseca et al., 2020). Finally, self-supervised learning and SimCLR (Chen et al., 2020b), a recent framework for contrastive self-supervision, are discussed.

3.1 Convolutional Neural Networks: A Primer

Convolutional networks, also known as convolutional neural networks (CNNs) or ConvNets, are deep neural networks that use the convolution operation in at least one of the layers instead of a regular feed-forward layer that uses general dense matrix multiplications. This section offers a quick overview of CNNs from the perspective of this thesis, first describing the convolution operation and comparing it in theory and practice, followed by CNNs and their presence in recent audio processing literature. This is followed up by a brief discussion of DeconvNets, a variety of CNNs that uses transposed convolutions and are most prominently known for their use in inspecting ConvNets, and are crucial to the proposed work.

3.1.1 What is a Convolution?

Convolution is a mathematical operation on two functions that accept a real valued argument, and is typically denoted with an asterisk (3.2).

$$\boldsymbol{s}(t) = \int \boldsymbol{x}(a)\boldsymbol{w}(t-a)da \tag{3.1}$$

$$\boldsymbol{s}(t) = (\boldsymbol{x} * \boldsymbol{w})(t) \tag{3.2}$$

In signal processing terminology, convolution can be simply seen as the sum of a signal *weighted* by a *kernel*. The discrete convolution operation with a discretized time index t is defined as:

$$\boldsymbol{s}(t) = (\boldsymbol{x} * \boldsymbol{w})(t) = \sum_{a=-\infty}^{\infty} \boldsymbol{x}(a)\boldsymbol{w}(t-a) \tag{3.3}$$

The first argument, $\boldsymbol{x}$ to the convolution is often referred to as the *input signal*, and the second argument $\boldsymbol{w}$ as the *kernel* or the *filter kernel*, which is adapted during the learning process. Since both the input signal and the kernel are stored as discrete-time signals on the computer and their values for non-stored indices are assumed to be zero, we can implement infinite summation as a finite sum over the array elements instead.

Therefore, for an n point input audio signal $\boldsymbol{x} \in \mathbb{R}^n$ and an m point kernel $\boldsymbol{w} \in \mathbb{R}^m$, the convolution operation yields an output signal $\boldsymbol{y}$ as follows:

$$\boldsymbol{y}(i) = (\boldsymbol{x} * \boldsymbol{w})(i) = \sum_{j=0}^{m} \boldsymbol{x}(j)\boldsymbol{w}(i-j) \tag{3.4}$$

This can be rewritten as Eq. 3.5 since convolution is commutative:

$$\boldsymbol{y}(i) = (\boldsymbol{w} * \boldsymbol{x})(i) = \sum_{j=0}^{m} \boldsymbol{x}(i-j)\boldsymbol{w}(j) \tag{3.5}$$

Convolution therefore is the weighted sum of the input with a sliding-window (the kernel) which is *flipped* left-to-right with respect to the input: the index into the input increases whereas it decreases into the kernel.

Cross-correlation and Convolution

In practice, most deep learning frameworks do not implement a proper convolution, but a closely related operation called *cross-correlation*.

$$\boldsymbol{y}(i) = (\boldsymbol{w} * \boldsymbol{x})(i) = \sum_{m} \boldsymbol{x}(i+m)\boldsymbol{w}(m) \tag{3.6}$$

Comparing Eq. 3.5 with 3.6, the difference is apparent: *cross-correlation* is convolution *without flipping the kernel.*

Cross-correlation and convolution are very different operations with conveniently similar mathematics. However, cross-correlation is slightly easier and more straightforward to implement, and since the kernel is learned on the fly on a case by case basis, the correlation based implementation will simply learn a flipped kernel in comparison to a strictly convolution based learning process. This is exactly the case with the "Conv1D" layer implemented in PyTorch (Paszke et al., 2019), which was used for this work.

It's worth pointing out that "Conv1D" implementations in popular deep learning frameworks are not causal by default. Causality in convolutions refers to the fact that the output at a given time step t should depend only on the inputs up to t. This however is beyond the scope of the proposed work. To read more on causality, readers should refer to Smith (1997); Lyons (2004).

Strided Convolutions

If the convolution operation can be viewed as a window that slides over the input and computes a weighted sum, the stride parameter governs how much the window "jumps" between each such computation. Strided convolutions are used for parametric (learned) downsampling, as opposed to pooling operations that downsample by using a summary statistic. Downsampling is useful for rapidly increasing the receptive field of a CNN and reducing computation load in the subsequent layers.

The stride parameter is not covered because strided convolutions are not used in the proposed approach, and the Max Pooling operation is used for downsampling feature maps. Interested readers should refer to Goodfellow et al. (2016) for more details about strided, as well as other variants of convolutions.

3.1.2 Pooling

In several instances where the presence of a feature is more important than the exact location, invariance to small local translations in the input is beneficial. Pooling operations do exactly that: by replacing the output of the neural net at certain positions with a summary statistic in the receptive field, pooling operations help make representations approximately invariant to small translations of the inputs. Paired with convolution layers, pooling layers force the neural net to learn local invariance (Goodfellow et al., 2016).

Pooling also allows us to summarize responses over very large neighbourhoods by simply reporting statistics every k timesteps. This effectively downsamples the feature map and reduces computation complexity for the following layer(s) in the neural net stack.

The most common pooling operations are Max Pooling (Zhou & Chellappa, 1988), which reports the maximum output, and Average Pooling, which reports the average output in the

receptive field of a sliding window.

Since our primary task is utterance-level sound event recognition (Section 3.3) where presence of an sound event is more important than the exact location, max pooling is used.

3.1.3 Max Un-pooling: Approximately Inverting Max Pooling

Although the max pooling operation is non-invertible, an approximate inverse can be obtained by simply storing the locations of the maximum outputs in a neighbourhood (called *switch indices*) during forward pass and restoring these max values at these particular locations, with the other positions remaining zero. Max Un-pooling (Dosovitskiy et al., 2015) has been used by several works to revert the pooling operation, ranging from convolutional Generative Adversarial Networks (GANs) (Goodfellow et al., 2020), convolutional autoencoders (Noh et al., 2015), to works on interpreting CNNs (Zeiler & Fergus, 2014).

3.1.4 CNNs in Machine learning for audio processing

CNNs are widely used across several tasks in the audio processing domain, such as speaker recognition (Nagrani et al., 2017; Yadav & Rai, 2018; Chung et al., 2018; Snyder et al., 2018; Xie et al., 2019; Yadav & Rai, 2020; Desplanques et al., 2020), automatic speech recognition (ASR) (Hannun et al., 2014; Amodei et al., 2016; Collobert et al., 2016; Li et al., 2019; Schneider et al., 2019; Baevski et al., 2020b; Gulati et al., 2020) and audio classification, to name a few.

In the speaker recognition domain, most of the recent and state-of-the-art approaches that use CNNs usually pair log-scaled spectrograms with prominent 2D-CNN architectures from the computer vision domain, essentially treating them like images (Nagrani et al., 2017; Yadav & Rai, 2018; Chung et al., 2018; Xie et al., 2019). CNNs have always been of interest for ASR, going back to the 1980s-90s (Waibel et al., 1989). CNN-RNN hybrid models, such as the DeepSpeech family (Hannun et al., 2014; Amodei et al., 2016), are also popular in the ASR domain, often paired with Connectionist Temporal Classification (CTC) loss (Graves et al., 2006) function to learn end-to-end acoustic models. More recently, several CNN only architectures for ASR have also emerged (Collobert et al., 2016; Li et al., 2019; Schneider et al., 2019; Baevski et al., 2020b). CNNs trained on spectrogram features are also popular in the audio classification domain (which includes sound event recognition and audio tagging) (Kong et al., 2020; Zeghidour et al., 2021; Chen et al., 2019b; Fonseca et al., 2020; Verma & Berger, 2021; Hyeji & Jihwan, 2019; Delphin-Poulat & Plapous, 2019; Lin et al., 2019; Kapka & Lewandowski, 2019; weon Jung et al., 2020; Chen et al., 2019a; Huang et al., 2019).

3.2 DeconvNets

Deconvolutional neural networks, also called DeconvNets, are a variant of CNNs that were originally proposed by Zeiler et al. (2011) for training hierarchical feature representations in a generative, unsupervised fashion. DeconvNets were later used by Zeiler & Fergus (2014) for interpreting intermediate feature representations of a trained ConvNet by mapping them back into the input pixel space, highlighting maximally activating substructures for a feature map of interest.

In Zeiler & Fergus (2014), DeconvNets use the deconvolution operation that shared weights with the convolutional layers of the trained CNN to inspect feature maps of interest. It's worth noting that the deconvolution operation in this context is kind of a misnomer and is not the same as the deconvolution operation in signal processing. Instead, deconvolution in a DeconvNet is a *transposed* convolution: a convolution operation with the filter axes transposed.

The proposed work utilizes DeconvNets to map intermediate feature representations of a raw waveform based 1D-CNN feature encoder back into the discrete-time audio input space, highlighting maximally activating substructures in the input, and is a crucial component of this study. To avoid reiteration of the same concepts, chapter 5 on Sonifications covers the DeconvNet based reconstruction process in detail.

3.3 What is Sound Event Recognition?

Recognizing and detecting acoustic events is a task of great importance in several domains, and is a key component of any successful human-computer interaction system that requires recognition of spoken language, as well as understanding acoustic scenes. Sound event recognition, also known as acoustic event recognition, is the task of recognizing the presence of acoustic events of interest in an audio clip. Sound event recognition problems present themselves as multiclass/multilabel classification problems with several acoustic events of interest spanning different audio sources, such as domestic appliances, musical instruments, or natural and urban environments.

Several datasets have been proposed for sound event recognition (Salamon et al., 2014; Piczak, 2015; Mesaros et al., 2016; Gemmeke et al., 2017; Fonseca et al., 2018, 2019; Turpault et al., 2019; Fonseca et al., 2020; Chen et al., 2020a). These range from very large-scale weakly labelled datasets like AudioSet (Gemmeke et al., 2017), manually annotated datasets like those from the Freesound (Fonseca et al., 2017) initiative, such as the FSDKaggle2018 (Fonseca et al., 2018) and FSD50K (Fonseca et al., 2020) datasets, and datasets collected using an automated pipeline multimodal pipeline, such as VGGSound (Chen et al., 2020a).

The primary dataset used in the current work is AudioSet (Gemmeke et al., 2017). AudioSet is a collection of ≈ 2.1 million weakly-labelled 10-second audio clips spanning 527 audio event

Subset	# Utterances	Duration (hrs)
Unbalanced train	1747465	4854
Balanced train	19253	53.48
Evaluation	17498	48.60

Table 3.1: AudioSet statistics

classes collected from YouTube videos, and is the largest audio event corpus openly available. AudioSet classes are organized in a hierarchy of labels originating from various sources, such as human sounds (eg. human voice, locomotion, group action), music, natural sounds (such as rustling of leaves, water streams), and objects (such as vehicles and explosions). However, the official release of AudioSet doesn't include the audio clips, and since all audio clips are sourced from YouTube videos, the complete corpus is not readily available due to geographical restrictions and original videos being removed. Table 3.1 shows statistics of AudioSet subsets and clips that were available and used.

Another sound event recognition dataset used in this study was FSD50K (Fonseca et al., 2020). FSD50K was used for benchmarking in-domain transfer learning performance of the proposed feature encoder, analyzing layerwise performance trends of contrastive and supervised pretraining, and finally for demonstrating that sonifications are maximally activating substructures in the input audio that are crucial for recognition performance. FSD50K is a manually annotated dataset of over 100 hours of audio data spanning 200 audio event classes, as shown in Table 3.2.

Subset	# Utterances	Duration (hrs)
Train	36796	70.40
Val	4170	10.35
Eval	10231	27.85

Table 3.2: FSD50K dataset statistics

3.4 Self-supervised learning

Deep neural networks are capable of learning very expressive feature hierarchies. However, they need a lot of labelled data to do so. Curating very high-quality hard-labelled data is very resource intensive, especially for the audio modality where labelling accurate event boundaries is difficult, and given the ever increasing complexity of neural architectures, is not feasible.

Self-supervised learning addresses this issue by exploiting the vast amounts of unlabelled data that is continuously being generated to learn feature representations that are transferable to downstream tasks. This is done by framing the *pretext task*, a supervised learning task formulated in such a way that partial information from an instance is used to predict the rest: and thus

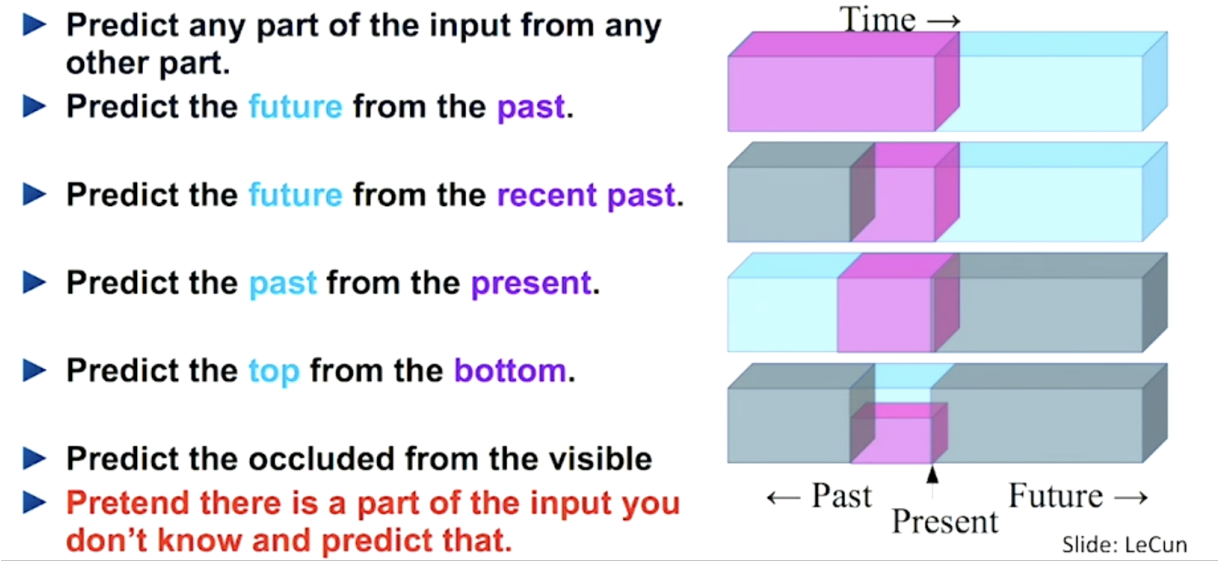

Figure 3.1: How to design pretext tasks (LeCun, 2018)

the *input* and the *target* are both available from the instance itself. A lot of recent works have proposed different pretext tasks and approaches to train self-supervised representations, falling primarily into generative (Donahue et al., 2016; Zhang et al., 2017b; Hu et al., 2020; Chung & Glass, 2020; Chung et al., 2019; Liu et al., 2021b; Chi et al., 2021; Ling et al., 2020), contrastive (Oord et al., 2018; Tian et al., 2020; He et al., 2020; Chen et al., 2020b; Grill et al., 2020; Qian et al., 2020) and predictive coding (Chung & Glass, 2020; Chung et al., 2020; Liu et al., 2021a; Hsu et al., 2021a) paradigms. Figure 3.1 depicts some sample pretext tasks.

In the audio and speech domain, the contrastive approach to self-supervised learning is quite popular (Oord et al., 2018; Baevski et al., 2020b; Schneider et al., 2019; Baevski et al., 2020a; Riviere et al., 2020; Hsu et al., 2021b; Kawakami et al., 2020; Baevski & Mohamed, 2020; Niizumi et al., 2021; Fonseca et al., 2021). The proposed work utilizes the SimCLR framework (Chen et al., 2020b,c) for contrastive self-supervised pretraining on the AudioSet dataset. SimCLR has already been demonstrated to work well in the sound event recognition domain by Fonseca et al. (2021); Saeed et al. (2021), and was found to be easier to implement and reproduce, unlike sequential modelling pretext task approaches (like (Oord et al., 2018)) which also do not perform well on AudioSet (Wang et al., 2021). The following section describes SimCLR in more detail.

3.4.1 The SimCLR way

The objective of the SimCLR approach is simple: learn representations that maximize agreement between different views of the same data sample. This is done by using a contrastive loss, termed *NT-Xent loss*, in the latent space. The following outlines the major components of the SimCLR

approach.

1. **Generate correlated views**: stochastic augmentation methods are used to generate different but correlated views of the same data sample, which will act as our positive pair denoted $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$.

2. **Encode using a base encoder**: A base encoder $f(.)$, which can be any network architecture, is used to obtain representation vectors $\boldsymbol{h}_i = f(\boldsymbol{x}_i), \boldsymbol{h}_i \in \mathbb{R}^d$ from our augmented samples.

3. **Project using projection head**: This representation $\boldsymbol{h}_i$ from base encoder is passed onto the projection head $g(.)$ that projects it on to the latent space $\boldsymbol{z}_i = g(\boldsymbol{h}_i) = g(f(\boldsymbol{x}_i))$. The projection head can be any DNN, but usually a small MLP is used.

4. **Apply NT-Xent loss on the projection**: Finally, NT-Xent loss is applied on the projection embedding $\boldsymbol{z}_i$. Given a set $\mathbb{S} = \{\tilde{\boldsymbol{x}}_k\}$ such that $\{\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j\} \subset \mathbb{S}$, the pretext task is to identify $\tilde{\boldsymbol{x}}_j$ in $\{\tilde{\boldsymbol{x}}_k\}_{k \neq i}$ for a given $\tilde{\boldsymbol{x}}_i$.

Given a random minibatch of N samples, the contrastive prediction task is defined on the $2N$ augmented samples derived from the minibatch (one pair per sample). Negative examples are not sampled separately, and given a positive pair, the other $2(N-1)$ augmented samples in a minibatch act as the negatives. Then, the *NT-Xent* Loss function (Sohn, 2016; Wu et al., 2018; Oord et al., 2018; Chen et al., 2020b) is defined as:

$$\ell_{ij} = -\log \frac{\exp\left(sim(z_i, z_j)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp\left(sim(z_i, z_k)/\tau\right)}, \tag{3.7}$$

where $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ are the representations for the positive, non-overlapping patches $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$, $\tau > 0$ is a temperature scaling, $\mathbb{1}_{v \neq i} \in \{0, 1\}$ is an indicator function that returns 1 if $v \neq i$, and N is the batch size. $sim(\boldsymbol{u}, \boldsymbol{v})$ is the cosine similarity between $\boldsymbol{u}$ and $\boldsymbol{v}$.

The SimCLR approach has several benefits:

- The freedom to use any reasonably parameterised base encoder and projection head without restrictions on the architecture.
- No need for computationally expensive negative sample mining.
- Usage of stochastic augmentations to generate correlated views.
- A memory bank (Wu et al., 2018; He et al., 2020) is no longer a necessity.

However, it is not without its disadvantages: training with extremely large batch sizes is needed for optimal performance, and finding the right combination of data augmentation methods is crucial for learning good representations (Chen et al., 2020b). Nevertheless, given the ease of

implementation and simplicity of the pipeline, its demonstrated effectiveness for SER tasks in existing literature, and above all, paired with the relatively lower compute requirements of our chosen feature encoder architecture, SimCLR was a straightforward choice.

Chapter 4

The setup: training and benchmarking

Several convolution neural network (CNN) architectures have been proposed for audio classification and sound event recognition (SER) tasks, ingesting a wide variety of input features, including 1D-CNN architectures trained on raw waveforms and log-spectrogram features. The majority of the current CNN based state-of-the-art methods utilize 2D-CNN architectures adapted from the computer vision domain trained on log-scaled spectrograms and other feature representations (Hershey et al., 2017; Fonseca et al., 2018; Hyeji & Jihwan, 2019; Turpault et al., 2019; Huang et al., 2019; Lin et al., 2019; Kapka & Lewandowski, 2019; Kong et al., 2020), including raw-waveform front-ends such as SincNet (Ravanelli & Bengio, 2018b) and LEAF (Zeghidour et al., 2021). However, the design choices behind developing a state-of-the-art neural architecture are not synergistic with the objectives of the proposed work, which is to establish a performant neural baseline that facilitates analysis of the proposed method of *sonification*.

This chapter focuses on deep neural network architecture design, training and performance benchmarking aspects of the proposed work in a way that is rigorous but at the same time does not lose sight of our main goal. Objectives and design choices behind *CNN12*, the 1D-CNN feature encoder at the heart of the proposed study, are discussed. This is followed by a detailed discussion of the evaluation protocol, which is composed of self-supervised and supervised learning experiments conducted on the AudioSet (Gemmeke et al., 2017) dataset and transfer learning experiments on various in-domain and out-of-domain downstream tasks to establish adequacy of the proposed CNN framework. Given the objective of this work, the performance benchmarks for each dataset do not reflect state-of-the-art performance (which would require considerable effort to determine and compare, in the first place), and only act as a measure of comparative performance versus recent comparable baselines.

4.1 CNN12, the proposed 1D-CNN feature encoder

Since our primary objective is to study feature representations learned by raw waveform based CNNs through sonification, the primary guiding principle is developing a feature encoder archi-

tecture that facilitates inversion of intermediate feature representations back into the audio input space while being simple to follow. This means that:

1. The feature encoder should not utilize specific raw waveform front ends (like LEAF and SincNet) and ingest raw signals directly so that the sonification process is decoupled from the front end used.

2. The feature encoder should strike a balance between simplicity and performance. A more sophisticated CNN paradigm, like ResNets (He et al., 2016) or DenseNets (Huang et al., 2017) make the sonification process more difficult.

This resulted in the following design decisions:

1. **Simple Sequential Stack of blocks**: The feature encoder should be a simple sequential stack of layer convolutional layer(s) followed by a non-linearity, one after the other.

2. **No Normalization Layers**: To keep the sonification process simple and independent from batch statistics, normalization layers such as Batch Normalization (Ioffe & Szegedy, 2015) or Instance Normalization Ulyanov et al. (2016), which are known to improve generalization performance, are not used. This has the added benefit of avoiding complexities associated with globally synchronized normalization techniques in distributed training, which is necessary to avoid local information leakage in contrastive models and learn better representations (Chen et al., 2020b).

The result is a simple 12-layer 1D-CNN feature encoder, named CNN12 (Figure 4.1). *CNN12* has 11 *conv-relu-[mp]* blocks, followed by temporal average pooling and a fully connected layer. It accepts a one-dimensional raw waveform signal $\boldsymbol{x} \in \mathbb{R}^T$, $\boldsymbol{x}_i \in \{-1, 1\}$ of T samples at a sampling frequency $F_s = 8000$ [Hz] and maps it into representative feature vector $\boldsymbol{h} = f_{enc}(\boldsymbol{x})$, $\boldsymbol{h} \in \mathbb{R}^{d=512}$. The sampling rate $F_s = 8000$ [Hz] was picked as it facilitated faster training and experimentation. We also benchmark an encoder that has 2x the number of filters in each convolutional layer, which we refer to as *CNN12*_x2 with the sole objective of evaluating how performance scales with added parameters for the given architecture, and if substantial, using it instead. All the core experiments on interpretability are conducted with *CNN12*.

4.2 Overview of the evaluation protocol

As mentioned earlier, the sole objective of this evaluation is to verify that the proposed architecture performs sufficiently well and is fit for evaluating the proposed interpretability method on contrastive self-supervised and supervised training paradigms. Given the complexity of training models on raw waveform inputs, an evaluation protocol spanning multiple audio datasets is adopted which covers performance of the proposed baseline representations across two sound

```
Encoder(
  (features): Sequential(
    (conv1): Conv1d(1, 64, kernel_size=(7,), stride=(1,))
    (act1): Activation(activation=relu, inplace=True)
    (mp1): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv1d(64, 64, kernel_size=(7,), stride=(1,))
    (act2): Activation(activation=relu, inplace=True)
    (mp2): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv3): Conv1d(64, 128, kernel_size=(5,), stride=(1,))
    (act3): Activation(activation=relu, inplace=True)
    (mp3): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv4): Conv1d(128, 128, kernel_size=(5,), stride=(1,))
    (act4): Activation(activation=relu, inplace=True)
    (mp4): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv5): Conv1d(128, 256, kernel_size=(5,), stride=(1,))
    (act5): Activation(activation=relu, inplace=True)
    (mp5): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv6): Conv1d(256, 256, kernel_size=(3,), stride=(1,))
    (act6): Activation(activation=relu, inplace=True)
    (mp6): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv7): Conv1d(256, 512, kernel_size=(3,), stride=(1,))
    (act7): Activation(activation=relu, inplace=True)
    (mp7): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv8): Conv1d(512, 512, kernel_size=(3,), stride=(1,))
    (act8): Activation(activation=relu, inplace=True)
    (mp8): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv9): Conv1d(512, 1024, kernel_size=(3,), stride=(1,))
    (act9): Activation(activation=relu, inplace=True)
    (conv10): Conv1d(1024, 1024, kernel_size=(3,), stride=(1,))
    (act10): Activation(activation=relu, inplace=True)
    (conv11): Conv1d(1024, 2048, kernel_size=(3,), stride=(1,))
    (act11): Activation(activation=relu, inplace=True)
  )
  (fc1): Sequential(
    (drop10): Dropout(p=0.4, inplace=False)
    (fc10): Linear(in_features=2048, out_features=512, bias=True)
  )
)
```

Figure 4.1: CNN12 feature encoder

event recognition tasks as well as transfer learning performance ont two out-of-domain tasks. Specifically,

1. AudioSet (Gemmeke et al., 2017) is used as our primary dataset for training contrastive and supervised representations for sound event recognition.

2. FSD50K (Fonseca et al., 2020) is used to evaluate how the primary AudioSet representations (both contrastive and supervised) performe when transferred to an in-domain sound event recognition task and compare how they stand with baselines trained from scratch.

3. Stopword recognition on the Speech Commands v2 (Warden, 2018) dataset and speaker identification on the VoxCeleb1 (Nagrani et al., 2017) dataset are used as proxy for evaluating transfer learning performance of the primary AudioSet representations on out-of-domain audio recognition tasks.

This protocol allows us to thoroughly evaluate whether the supervised and contrastive representations trained on AudioSet are suitable and achieve satisfactory performance by comparing them with recent, and relevant existing baselines trained on raw waveform signals. The following sections cover each step in more detail, discussing performance metrics used as well as the benchmark results.

4.3 Contrastive self-supervised pretraining using SimCLR on AudioSet

As previously mentioned, we use the SimCLR framework (Chen et al., 2020b), which has been discussed in-depth in Section 3.4.1, for training contrastive self-supervised feature representations. The proposed *CNN12* feature encoder is used as the base encoder in our SimCLR pipeline. The following subsections cover other details related to the SimCLR based contrastive pretraining in this work.

4.3.1 Generating correlated segments from a raw waveform sample

Generating correlated "views" of the same input sample is critical for training contrastive representations that perform well (Chen et al., 2020b), and the composition of data augmentations used significantly affects quality of representations learned.

Given an input audio signal $\boldsymbol{x} \in \mathbb{R}^T$, two non-overlapping crops $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, each 2.5 seconds in duration, are randomly cropped from the input signal. These samples are then independently fed to the stochastic augmentation module to generate correlated segments, which is comprised of the following augmentations:

- Random Gain perturbation: The gain of the non-overlapping crops is randomly perturbed in the $[-18, 6]$ dB range, with a probability of 0.5.
- Additive Gaussian Noise: Random gaussian noise with a magnitude in the $[0.001, 0.015]$ range is added with a probability of 0.5.
- Random Timestep masking: Inspired by SpecAugment (Park et al., 2019), up to 2 masks in the time domain covering up to 500 ms duration each were applied randomly to the input segment.
- Background injection: Finally, random samples from the AudioSet were injected to 50% of the input crops as background noise while preserving a randomly selected signal-to-noise (SNR) ratio between $[10, 25]$. This augmentation method is inspired by and similar to the *mix-back* augmentation utilized by Fonseca et al. (2021) for self-supervised pre-training using SimCLR on a noisy dataset.

The performance of these methods, along with other relevant SimCLR hyperparameters, is given in Table 4.1 in accordance with the evaluation methodology highlighted in Sec 4.3.3.

4.3.2 Projection head for SimCLR based pretraining

In the SimCLR framework, a projection head is used on top of the base encoder to project features into an embedding space (Section 3.4.1). A multilayer perceptron (MLP) with one-hidden layer and 512 hidden units is used as the projection head $f_{proj}(.)$ to project features $\boldsymbol{h} = f_{enc}(x)$ from the *CNN12* feature encoder on an embedding space $\boldsymbol{z} = f_{proj}(f_{enc}(\boldsymbol{x})), \boldsymbol{z} \in \mathbb{R}^{d=512}$, on which *NT-Xent* loss (Chen et al., 2020b) is used.

4.3.3 Putting it all together

Putting everything together, we train the contrastive representations by minimizing the following objective function:

$$\ell_{ij} = -\log \frac{\exp\left(sim(z_i, z_j)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp\left(sim(z_i, z_k)/\tau\right)}, \tag{4.1}$$

where $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ are the representations for the positive, non-overlapping patches $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, with other parameters as discussed in Sec 3.4.1.

In line with self-supervised learning literature (Oord et al., 2018; He et al., 2020; Grill et al., 2020; Tian et al., 2020; Chung et al., 2020; Baevski et al., 2020b; Fonseca et al., 2021), linear evaluation, also called linear probing, was used to quantify contrastive model performance. Linear evaluation consists of training a classifier on top of the features extracted from a "frozen" feature encoder. A random holdout set consisting of 10% of the "unbalanced train" set was used

Setting	τ	fconly (*mAP*)	full (*mAP*)
CNN12 with randomgain	0.1	0.112 ± 0.002	0.203 ± 0.001
+ gaussiannoise	0.1	0.142 ± 0.001	0.210 ± 0.003
+ timemask	0.1	0.142 ± 0.001	0.211 ± 0.002
+ random SNR	0.1	0.148 ± 0.003	0.215 ± 0.001
CNN12 (all)	0.2	0.120 ± 0.001	0.205 ± 0.002
CNN12 (all)	0.3	0.105 ± 0.001	0.200 ± 0.001
CNN12_x2 (all)	0.1	0.151 ± 0.003	0.217 ± 0.005

Table 4.1: AudioSet Contrastive pretraining linear eval performance for various hyperparameters when trained on the *balanced train* subset. *fconly* represents linear-eval, whereas full represents finetuning the entire model. 95% confidence intervals are reported over several runs.

for tuning basic hyperparameters, such as batch size, learning rate and learning rate schedule. For the reported experiments, Adam optimizer (Kingma & Ba, 2015) with default learning rate and a weight decay of $1e-3$ was used with a linear warmup for the first 10 epochs followed by a cosine decay schedule without restarts (Loshchilov & Hutter, 2016). All experiments were done on a TPUv3-8 machine, with a batch size of 2048, all contrastive experiments use $\tau = 0.1$, unless stated otherwise.

To evaluate SimCLR related hyperparameters, including the various augmentations used, linear evaluation was done using the "balanced train" subset of AudioSet, and mean average precision with macro-averaging (*mAP*) was used as the performance metric. Table 4.1 shows how stronger augmentation leads to better recognition performance, which is inline with observations made by Chen et al. (2020b). For our case, the default temperature parameter value of $\tau = 0.1$ as used by Chen et al. (2020b) performs the best. However, performance does not scale well with number of parameters for the large feature encoder, CNN12_x2, hinting at possible overfitting on the small "balanced train" subset. Models trained with the settings described by Row 4 in Table 4.1 are our contrastive baseline for the rest of the work.

4.4 Supervised learning experiments

This section focuses on supervised learning experiments conducted on all the datasets in consideration. For supervised learning experiments, a fully connected layer is added on top of the *CNN12* feature encoder. The model is then trained by solving a classification loss depending on the dataset and task in consideration: multilabel binary crossentropy is used for both AudioSet and FSD50K, whereas multiclass categorical cross-entropy is used for both Speech Commands v2 and VoxCeleb1 datasets. Apart from experiments that require training the entire model on the entire AudioSet dataset, all supervised learning experiments were done on a single tpu-core of a TPUv3-8 machine using Adam optimizer with a linear warmup for 10 epochs followed by

Model	Pretraining	*fconly*	mAP	mAUC	dprime
CNN12	False	NA	0.299 ± 0.001	0.954 ± 0.000	2.378 ± 0.007
CNN12	True	True	0.197 ± 0.003	0.918 ± 0.006	1.966 ± 0.057
CNN12	True	False	0.327 ± 0.009	0.958 ± 0.001	2.436 ± 0.011
CNN12_x2	True	False	0.332 ± 0.005	0.959 ± 0.001	2.438 ± 0.020

Table 4.2: AudioSet Supervised Learning performance. Pretraining indicates whether contrastive pretrained weights were used

Model	# Params	F_s	features	mAP	mAUC	dprime
CNN14 (Kong et al., 2020)	80M	16 kHz	log-melspec	0.431	0.973	2.732
CNN14 (Kong et al., 2020)	80M	8 kHz	log-melspec	0.406	0.97	2.654
CNN14 (Zeghidour et al., 2021)	80M	16 kHz	SincNet	-	0.97	2.66
CNN14 (Zeghidour et al., 2021)	80M	16 kHz	LEAF	-	0.974	2.74
CNN14 (Zeghidour et al., 2021)	80M	16 kHz	Wavegram	-	0.961	2.5
1D+2D ResNet (Chen et al., 2019b)	-	16 kHz	raw signal	0.372	0.968	2.614
CNN12	14M	8 kHz	raw signal	0.323	0.957	2.436
CNN12_x2	53M	8 kHz	raw signal	0.332	0.959	2.438

Table 4.3: AudioSet SER performance with comparable baselines. Confidence intervals omitted.

a cosine decay. Random gain, additive gaussian noise, random time masking and mixup (Zhang et al., 2017a) augmentations were used for all supervised models. No rebalancing of classes is done for any of the datasets, and all evaluation and metrics calculation was done on the utterance level, ingesting the entire audio clip at once. Each experiment is repeated at least 3 times, and 95% confidence intervals are reported, The following sections discuss each of these experiments and their outcomes in detail.

4.4.1 Supervised learning on AudioSet

For AudioSet supervised experiments, training was done on 5-sec random crops, with a 10% random holdout set used for setting hyper-parameters. Once done, all reported models were trained on the entire AudioSet training set, which includes both the "unbalanced train" and the "balanced train" subsets (as described in Section 3.3 and Table 3.1), and mean average precision (*mAP*), mean AUC score (*mAUC*) and *d-prime* metrics calculated on an utterance level prediction (entire audio clip) were reported on the evaluation set, which is inline with training protocol widely adopted in literature. A per-TPU-core batch size of 128 was used, which equates to an effective batch size of 1024 for experiments on the entire AudioSet dataset.

Table 4.2 lists the results of the experiments conducted under various settings under the proposed work. The *pretraining* flag indicates whether pretrained weights from the contrastive baseline setup in Sec 4.3.3 were used, whereas the *fconly* flag indicates linear evaluation or fine-tuning the entire model. It is evident that contrastive pretraining yields substantial performance

improvement over supervised only baseline (row 1). It is worth noting that CNN12_x2 continues to scale poorly with number of parameters: as a result, we decided to stick with CNN12 for the majority of the core experiments.

Table 4.3 compares performance of the proposed approach with comparable baselines. This list is non-exhaustive, and is not indicative of state-of-the-art performance on the AudioSet benchmark, but given how our objective is not to challenge the state-of-the-art, it's a good representation of indicative performance.

4.4.2 In-domain transfer: FSD50K

Model	Pretraining	*fconly*	mAP	mAUC	dprime
CNN12	None	NA	0.362 ± 0.006	0.909 ± 0.001	1.889 ± 0.010
CNN12	Contrastive	True	0.306 ± 0.007	0.873 ± 0.003	1.611 ± 0.024
CNN12	Contrastive	False	0.424 ± 0.004	0.921 ± 0.001	2.001 ± 0.006
CNN12	Supervised	True	0.463 ± 0.002	0.935 ± 0.001	2.144 ± 0.015
CNN12	Supervised	False	0.448 ± 0.004	0.928 ± 0.003	2.068 ± 0.028
CNN12_x2	Contrastive	True	0.307 ± 0.006	0.869 ± 0.004	1.584 ± 0.020
CNN12_x2	Contrastive	False	0.432 ± 0.003	0.924 ± 0.001	2.032 ± 0.004

Table 4.4: FSD50K eval performance. Pretraining refers to type of pretraining on AudioSet

Model	#Params	F_s	features	mAP	dprime
CRNN (Fonseca et al., 2020)	0.96M	22.05 kHz	log-melspec	0.417	2.068
VGG-like (Fonseca et al., 2020)	0.27M	22.05 kHz	log-melspec	0.434	2.167
ResNet-18 (Fonseca et al., 2020)	11.3M	22.05 kHz	log-melspec	0.373	1.883
DenseNet-121 (Fonseca et al., 2020)	12.5M	22.05 kHz	log-melspec	0.425	2.112
Large Transformer (Verma & Berger, 2021)	2.3M	16 kHz	raw signal	0.537	-
CNN12 (supervised-*fconly*)	14M	8 kHz	raw signal	0.463	2.144
CNN12 (contrastive)	14M	8 kHz	raw signal	0.424	2.001

Table 4.5: FSD50k eval performance vs comparable baselines. Confidence intervals omitted.

As opposed to AudioSet, the FSD50K dataset consists of audio clips of varying duration, with the average clip shorter than 5 seconds. Training was done on 2.5 sec random crops, and the validation set (Section 3.3, Table 3.2) was used for hyper-parameter tuning. Training was done on a single core of a TPU-v3 device with a batch size of 64. Inline with previous baselines, mAP and *dprime* metrics are reported on utterance level prediction.

Table 4.4 lists results of all experiments conducted in the proposed work. The *pretraining* parameter indicates whether supervised or contrastively pretrained baseline model was finetuned. It is worth noting how both contrastive and supervised pretraining improve eval performance drastically over the baseline FSD50K model (row 1), with the latter being better.

Subset	# Utterances	Duration (hrs)
training	84843	23.11
validation	9981	2.72
testing	11005	3.00

Table 4.6: Speech Commands v2 statistics

FSD50K is a relatively new benchmark, and as a result there are fewer baselines available. Table 4.5 compares performance of the two best performing proposed methods (bold) with existing models.

4.4.3 Out-of-domain transfer #1: Stopwords recognition on Speech Commands v2

Speech Commands v2 (Warden, 2018) is a dataset of stopword recognition. It consists of 105829 audio clips, each 1 sec in duration, split into *training*, *validation* and *testing* sets. Each sample consists of a spoken utterance of a single stopword out of a total of 35 (Table 4.6).

Speech Commands v2 presents a multiclass classification task. Since all audio samples are $\leq$ 1 sec in duration and consist of a single stopword, no random cropping was done while training. A batch size of 128 was used for all experiments, and classification accuracy on an utterance level is used.

Table 4.7 reports classification accuracy of the proposed feature encoder, and as a measure of indicative performance, 2 recent baselines that are also trained on raw waveform signals (albeit with a designated front-end) are reported. As opposed to FSD50K, which was an in-domain task and benefitted from recognition specific features learned by the supervised pretrained model, contrastive pretraining achieves the best performance. This is expected behaviour, as contrastive representations are supposed to be more generic and widely applicable.

Model	F_s	features	Accuracy%
EfficientNet-B0 (Saeed et al., 2021)	16 kHz	log-mel	95.5
EfficientNet-B0 (Zeghidour et al., 2021)	16 kHz	LEAF	93.4
CNN12	8 kHz	raw signal	92.3 ± 0.5
CNN12 (Supervised-*fconly*)	8 kHz	raw signal	36.0 ± 1.4
CNN12 (Supervised-*full*)	8 kHz	raw signal	92.9 ± 0.7
CNN12 (Contrastive-*fconly*)	8 kHz	raw signal	29.5 ± 0.6
CNN12 (Contrastive-*full*)	8 kHz	raw signal	93.4 ± 0.7

Table 4.7: Speech Commands v2 Test Accuracy. Recent comparable baselines for reference. Paranthesis indicate pretrained weights used in the proposed work (bold)

4.4.4 Out-of-domain transfer #2: Speaker identification on VoxCeleb1

VoxCeleb1 (Nagrani et al., 2017) is a dataset for text-independent speaker identification and verification in the wild. The speaker identification subset consists of 153516 audio files of varying duration spanning 1251 speakers of various nationalities, ethnicities and speaking several languages. Several recent works that model raw waveform signals and propose front-end models for raw waveform processing have used this dataset for benchmarking.

Subset	# Utterances	Duration (hrs)
training	138361	318.41
validation	6904	15.18
testing	8251	18.06

Table 4.8: VoxCeleb1 Speaker Identification subset statistics

For VoxCeleb1 speaker identification benchmarking, models were trained on random 2.5 sec crops using categorical cross-entropy. A batch size of 128 was used for all experiments, and utterance level classification accuracy is reported.

Model	F_s	features	Accuracy%
EfficientNet-B0 (Zeghidour et al., 2021)	16 kHz	SincNet	43.5
EfficientNet-B0 (Zeghidour et al., 2021)	16 kHz	LEAF	33.1
CNN12	8 kHz	raw signal	15.53 ± 0.5
CNN12 (Contrastive-*fconly*)	8 kHz	raw signal	20.45 ± 0.1
CNN12 (Contrastive-*full*)	8 kHz	raw signal	32.46 ± 0.2

Table 4.9: VoxCeleb1 Speaker Identification Test Accuracy. Recent comparable baselines for reference. Paranthesis indicate pretrained weights used in the proposed work (bold)

Table 4.9 reports classification accuracy of the proposed methods with comparable recent baselines. SincNet features were proposed specifically for speaker recognition, as evidenced by their extremely strong performance. The proposed approach with contrastive pretraining achieves comparable performance with a comparable recent raw waveform baseline, despite being trained on lower sampling rate of 8000 Hz.

4.5 Closing Comments

Based on the previous sections and benchmark results, it's evident that while the proposed approach does not achieve state-of-the-art performance, it achieves satisfactory performance when compared to relevant, recent baselines. The proposed contrastive and supervised CNN12 feature encoder is thus believed to be fit for for the purpose of the study and sufficiently well performing for intepretability analysis using sonifications.

Chapter 5

Sonification

The chapter presents the primary proposal of this work: interpreting and comparing contrastive self-supervised and supervised CNNs by inverting intermediate representations back into the discrete-time audio signal input space, a process appropriately named *sonification*. The following sections define what sonifications are and the process of sonification for the given *CNN12* feature encoder. These are followed by general observations made regarding contrastive and supervised representations by listening to the sonifications as well as visualizing them, as well as how sonifications can be used for interpreting training time evolution of features.

5.1 Definition

The term *sonification* refers to the process of mapping feature maps from intermediate layers of the proposed *CNN12* feature encoder back into the discrete-time audio input space where they can be listened to as well as visualized in the frequency domain. This process can be seen as the audio processing counterpart of the *deep visualization* process prominently utilized for interpreting image recognition CNNs in the computer vision domain (Erhan et al., 2009; Simonyan et al., 2013; Zeiler & Fergus, 2014; Springenberg et al., 2015; Mahendran & Vedaldi, 2015; Smilkov et al., 2017; Selvaraju et al., 2020).

Sonifications refer to the interpretations made using the sonification process, similar to how the interpretations of the deep visualization process are simply referred to as visualizations. Therefore, for a particular raw waveform input $\boldsymbol{x} \in \mathbb{R}^T$, sonifications corresponding to a particular feature map of interest will be a discrete-time signal $\boldsymbol{s} \in \mathbb{R}^T$ in the audio input space.

5.2 How Sonification works

In this work, we utilize DeconvNets (Zeiler et al., 2011; Zeiler & Fergus, 2014) for sonification. DeconvNets use transposed convolutions, better known as deconvolutions in the deep learning literature, for approximately inverting the convolution filtering done by a layer in a

traditional convolutional neural network (CNN). The transposed convolutions of a DeconvNet share weights with the convolutional layer they are invert, and require no additional training.

5.2.1 The sonification process

Given an input waveform $\boldsymbol{x} \in \mathbb{R}^T$ with T samples and a trained CNN12 feature encoder $f_{enc}(.)$, let $L \in \{1, 11\}$ denote the *conv-relu-[mp]* block whose *conv* layer has f_{out} total number of filters. Let $f_{L_i} \in \{1, f_{out}\}$ be the index of the feature map in layer L we want to inspect, and $\boldsymbol{s} \in \mathbb{R}^T$ be the corresponding sonification output. Then, the sonification process can be explained as follows:

1. **Forward Pass:** Execute forward pass on the CNN12 feature encoder, storing all intermediate feature maps and switch indices.

2. **Starting feature map of interest:** Starting with intermediate feature maps for the block of interest L, all activations in layer L except f_{L_i} are set to zero, and the resulting features are passed on to the corresponding deconvnet for reconstruction.

3. **Applying the deconvnet**

   i. 1D-MaxUnpooling: Depending on the block of interest, a 1D-MaxUnpooling operation (Dosovitskiy et al., 2015) is applied on the input feature maps to revert the MaxPool operation. MaxUnpooling utilizes switch indices, which are locations of the max element recorded during MaxPooling. We observed that using a strided convolution for downsampling in the feature encoder distorts the time-domain structure of the sonified waveform and results in significantly worse sonifications. In contrast, MaxPooling and MaxUnpooling preserve *time-invariance* of input signal and the sonification, i.e. for input waveform $\boldsymbol{x}$, and the corresponding sonification w.r.t. a feature map $\boldsymbol{s}$, any integral offset k in input $\boldsymbol{x}$, i.e. $\boldsymbol{x}[n+k]$, results in equal offset $\boldsymbol{s}[n+k]$ in the corresponding sonification (see Figure 5.1). This is expected, as the only downsampling operation in the feature encoder is reverted by max-unpooling, which preserves input structure by using switches to place reconstructions at the correct locations.

   ii. Rectification: ReLU activation is applied to the reconstructed signal since all convolution layers are accustomed to receiving positive feature maps.

   iii. Transposed convolution, a.k.a deconvolution: This step is an approximate inverse of the corresponding convolutional layer in the proposed feature encoder, reverting the feature maps into outputs from the previous block. Transposed convolutions share weights with the convolutional layer they invert, and require no additional training.

   The above steps are repeated until we reach the input signal space, yielding the output sonification signal $\boldsymbol{s}$.

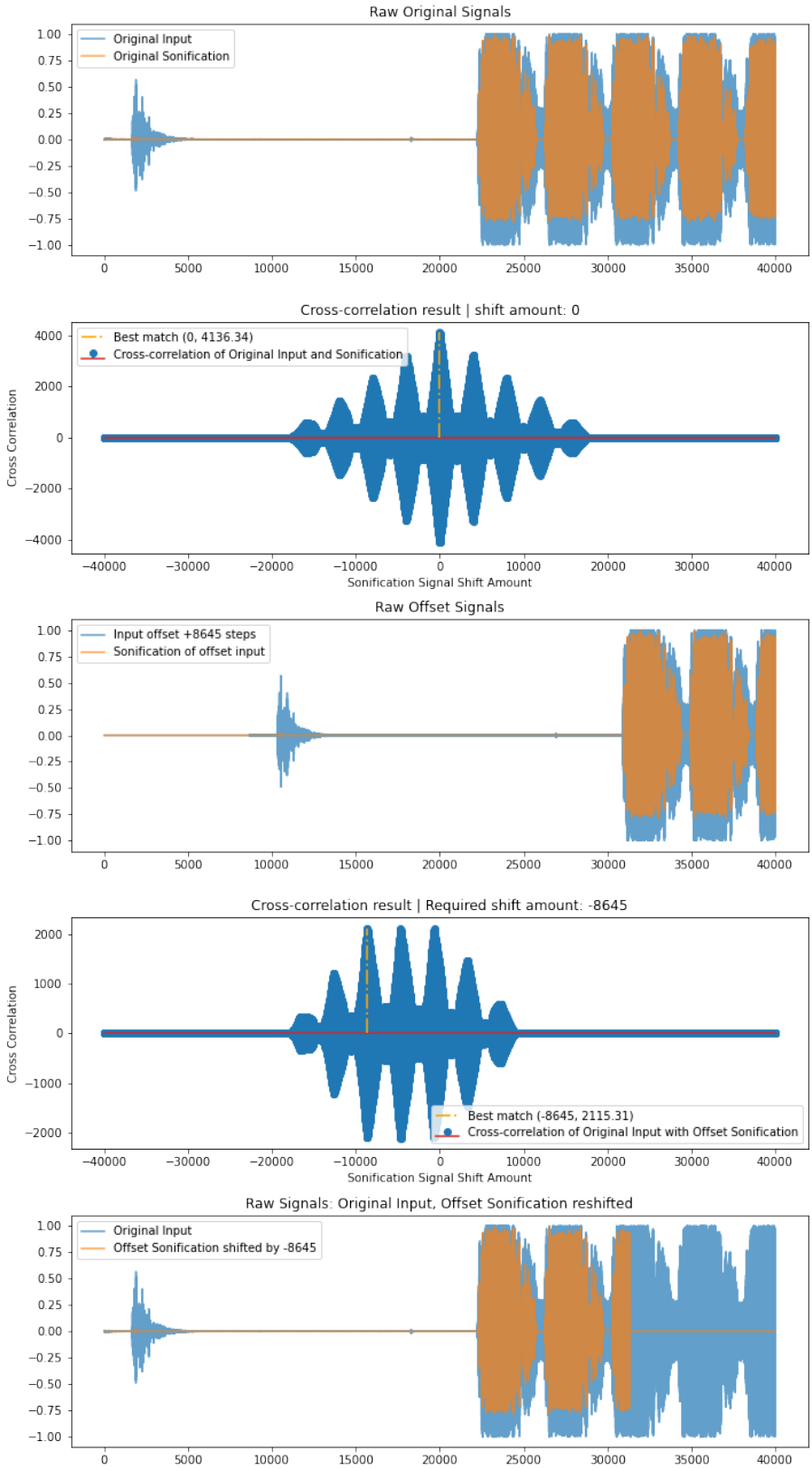

Figure 5.1: Time-invariance of the input and corresponding sonifications. In the figure, sonification of an input offset by (randomly selected) +8645 steps is centered with the original input by left-shifting it by the same amount. Other samples demonstrated the same

4. **Post-processing:** The output sonification signal $\boldsymbol{s}$ is then scaled to have a unit peak magnitude, followed by multiplication with the maximum magnitude of the input waveform, yielding $\boldsymbol{s} \in \{-\max|\boldsymbol{x}|, \max|\boldsymbol{x}|\}$.

The sonification algorithm using DeconvNets is shown in Algorithm 1.

Algorithm 1 The sonification algorithm, where f_{enc} is the CNN12 feature encoder, *deconvnets* is a dictionary of encoder *layer:DeconvNet* pairs, x is the input signal, $L \in [1, 11]$ is the CNN12 block of interest and m is the feature map in block L to inspect

procedure SONIFICATION($f_{enc}, deconvnets, x, L, m$)
  $f_{maps}, s_{indices} = f_{enc}(x)$ ▷ where $f_{maps}, s_{indices}$ are named key value pairs
  temp = f_{maps}[L]
  currentMap = ZEROSLIKE(*temp*)
  currentMap[m] = temp[m]
  currBlock = L
  while *currBlock* ≥ 1 **do**
    D = deconvnets[currBlock]
    if $s_{indices}$[*currBlock*] **then**
      currentMap = D.UNPOOL(currentMap, $s_{indices}$[currBlock])
    end if
    currentMap = RELU(currentMap)
    currentMap = D.TRANSPOSEDCONV(currentMap)
    currBlock -= 1
  end while
  s = currentMap ▷ $s \in R^T$ is the sonified signal in the input space
  s = MAXABSSCALE(s, min=-1, max=1)
  s $*=$ max(abs(x)) ▷ s is scaled to $\{-max|x|, max|x|\}$
  return *s*
end procedure

Following are some noteworthy keypoints regarding the proposed sonification process:

- The process of applying DeconvNets for sonification is non-parametric in nature, since transposed convolutions share weights with the convolution filter they invert and no additional training process is involved (Zeiler & Fergus, 2014).

- The sonifications generated are not *samples* from the DeconvNet since there is no generative process involved. Sonifications represent substructures in the input that maximally activate a given feature map: as demonstrated in Section 6.2, removing these sonifications from the input signals significantly alters feature map activity resulting in performance degradation.

- Using DeconvNets for sonification only allows interpreting a single feature activation and not the cumulative activity of the entire layer; attempts made to sonify the cumulative activity in a layer yielded poor quality sonifications. This is also highlighted in Zeiler & Fergus (2014).

5.3 Observations

Sonifications can be directly listened to as well as inspected visually in the frequency domain as log-scaled spectrograms. Dedicated Weights and Biases (Biewald, 2020) dashboards for contrastive[1] and supervised representations[2] were used for inspection (PS: these dashboards contain select samples, as uploading all the data results in significant slowdown). Sonifications corresponding to the top-5 maximally activating inputs for each feature map of each layer of the *CNN12* feature encoder were manually inspected, for both the contrastive and supervised baseline AudioSet models highlighted in Chapter 4.

The following subsections discuss core observations made from inspecting sonifications. Figures with select input-sonification pairs from select feature maps that justify the observations made are provided on a case-by-case basis. Larger figures with more samples that visualize feature maps from all layers can be found in Appendix A. This is followed by visualization of train time feature evolution in contrastive and supervised *CNN12* models.

5.3.1 Band Selectivity and Feature Map Activity

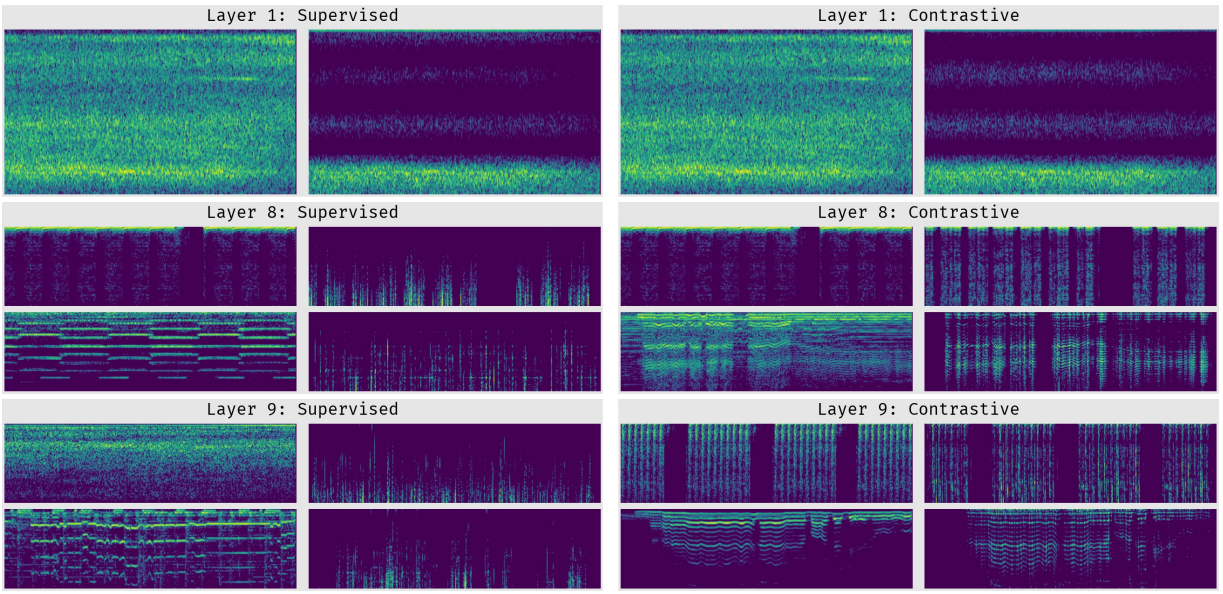

Figure 5.2: Comparing band selectivity and feature map activity across supervised and contrastive representations by visualising input-sonification pairs as log-scaled spectrograms. Note the greater activity in the higher frequency bins of the contrastive representations for layer 8 and 9.

Both supervised and contrastive representations learn simple band-selective filters in the initial layers. Supervised learning is generally more band-selective and has less active feature maps

[1] https://wandb.ai/sarthakyadav/select-sonifications-supervised
[2] https://wandb.ai/sarthakyadav/select-sonifications-contrastive

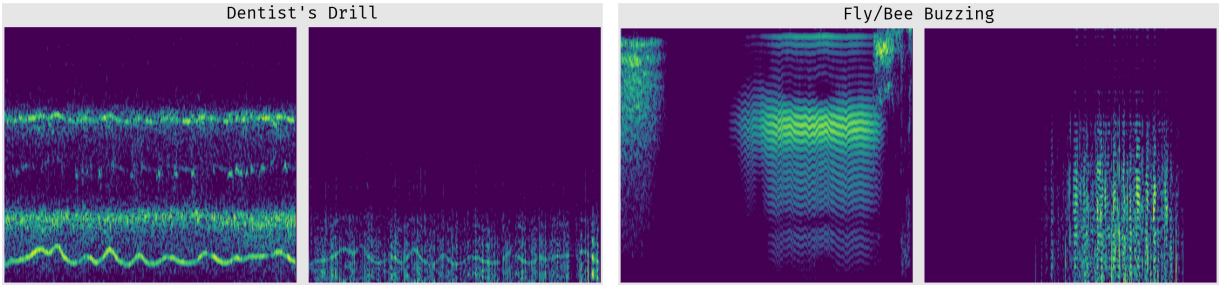

Figure 5.3: Select supervised input-sonification pairs. For supervised models, most remarkable sonifications were found in the middle layers (layers 3-7).

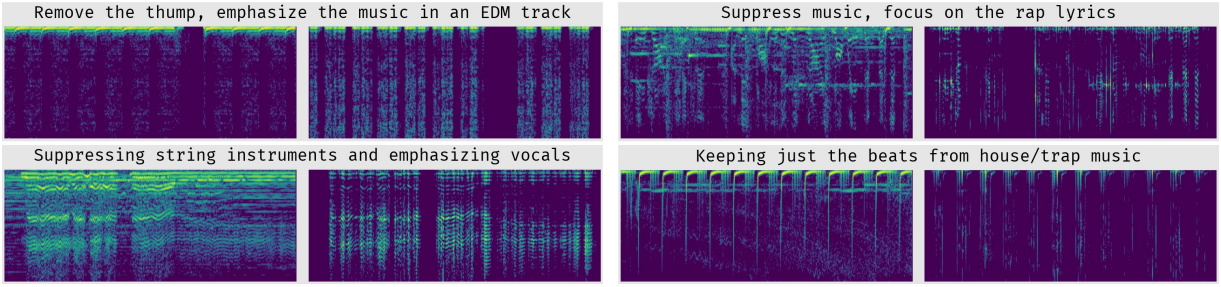

Figure 5.4: Select contrastive input-sonification pairs. These samples are from layers 8 and 11, and similarly remarkable sonifications can be found throughout all the layers.

in the deeper layers, a notion well supported by conventional wisdom: deeper layers learn more specialized concepts. However, contrastive representations appear to be different than supervised representations in this aspect. Figure 5.2 shows input-sonification pairs obtained from Layer 1, Layer 8 and Layer 9 of the feature encoder trained under the two regimes. It's worth noting how band selectivity and feature map activity is similar in the first layer across the two training paradigms, whereas deeper contrastive representations (layer 8 and 9) are comparatively more active. Similar general trends were observed while evaluating other examples.

5.3.2 Contrastive representations yield more acoustically discernible sonifications

For the supervised model, we observed that most of the acoustically remarkable sonifications occur in layers 3-7, such as a dentist's drill or the buzzing of a fly/bee (See Figure 5.3). As we go deeper, sonifications become less acoustically remarkable and more noisy: sonifications from layers 8-11 consist mostly of high frequency noise and are not acoustically discernible. Although it is difficult to say for certain without further exploration, there are several plausible explanations:

- A simple explanation could be that reconstruction simply becomes more difficult the farther we are from the input layer.
- We suspect that max-unpooling might also contribute to this phenomena: it simply replaces maximum values at the correct indices and leaves other elements zero, possibly inducing discontinuities and contributing to noise.
- Finally, it can simply be the nature of supervised representations itself. Deeper layers might be emphasising the presence of discriminative cues that, although crucial for recognition performance (Figure 6.1), do not correspond to intelligible acoustic events in the inputs, but merely indicate the presence of them.

However, we observe that contrastive representations do not follow the same trend, or at least not to the same extent, an observation supported by experiments done in Section 6.3. Contrastive representations from the deeper layers result in sonifications that are significantly more acoustically discernible. We discovered remarkable sonifications as deep as layers 7-11, such as news presenters, a feature map that removes heavy "thumps" from an EDM track, a feature map that removes background music but keeps the rap lyrics, a feature map that removes instruments but keeps the vocalizations in a folk song, and a feature map that suppresses house/trap music but keeps the beats (see Figure 5.4). This indicates that the properties of supervised representations could indeed be the major contributing factor behind the lower reconstruction quality and noise in the sonifications of deeper layers, and needs further exploration.

We also observed that contrastive representations demonstrate stronger acoustic content coupling in the deeper layers in comparison to supervised representations. For example, we observed that a large number of feature maps in layer 11 modelled music and human-vocalization related information, whereas sonifications from layer 11 in the supervised representations modelled a wider variety of maximally activating inputs, ranging from simple acoustic elements such as alarms and bells to complex ones, such as music and speech.

5.3.3 Using sonifications to visualize training time feature evolution

Sonifications can also be used to study how features evolve during training time, visualized as log-scaled spectrograms of the sonifications obtained from randomly selected feature maps and their corresponding maximally activating inputs from the AudioSet dataset.

The process is as follows: the maximally activating input in the AudioSet evaluation set are determined for a set of randomly selected feature maps in the final contrastive model checkpoint. Using these feature map indices and their corresponding input waveforms, we *go back in training time*, performing sonifications for each feature map of interest. Doing this yields a smoother visualization of training time feature progression than Zeiler & Fergus (2014), where

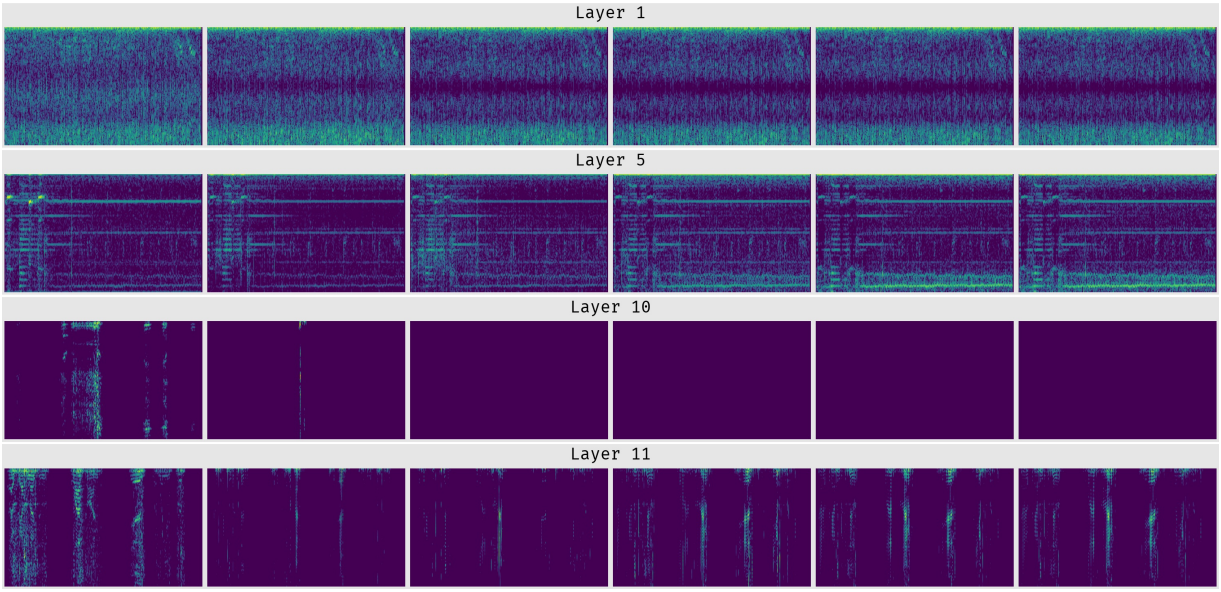

Figure 5.5: Contrastive feature evolution in the select feature maps from several layers. Note how certain frequency bins are being increasingly emphasised in the layer 5 feature map, while dead feature map in layer 10 starts with a full spectrogram, but fades away later.

sudden jumps appeared in the visualization whenever the maximally activating input changed. This process is then repeated for the supervised model with no contrastive pretraining.

Figure 5.5 shows training time feature evolution for some feature maps. For several feature maps, sonifications at the first checkpoint show a full spectrogram, even for the dead units, which changes over training time. More comprehensive visualizations of feature maps from all layers of the supervised and the contrastive models can be found in Appendix A.

This evaluation can ideally act as an indicator for the cause of dead units in a layer; whether they were caused by a suboptimal initialization scheme or just excessive number of filters in a layer. However, more thorough evaluation is necessary for this claim, which was not conducted in this work.

Chapter 6

Experiments and analysis

This chapter covers experiments conducted with the intention of supporting observations made in Chapter 5 as well as to highlight differences between contrastive and supervised representations using sonifications.

First, a layerwise analysis of transfer learning performance is conducted on FSD50K and Speech Commands v2 datasets, followed up by an experiment that verifies whether sonifications actually stimulate their respective feature maps using a residual signal study. Finally the similarity between sonifications and their corresponding inputs is quantified using spectral analysis and is used to compare contrastive and supervised representations, supporting observations made in Section 5.3.

6.1 Analyzing layer importances

6.1.1 Objective

Upon inspection of the sonifications in Section 5.3 it was discovered that most supervised sonifications displayed very low feature map activity in the deep layers, which raised suspicion regarding quality of the representations learned. We suspected that the layers were only marginally contributing to recognition performance, and whether the proposed architecture was improperly initialized, possibly leading to dead neurons which was instead perceived as low feature map activity. The objective of this experiment was to rule out this suspicion and check how these layers contributed to recognition performance.

6.1.2 Methodology

To explore how different layers from the contrastive and supervised models contribute to recognition performance, we utilize FSD50K and Speech Commands v2 dataset, as they represent an in-domain and an out-of-domain task, as well as faster experimentation, since these are orders

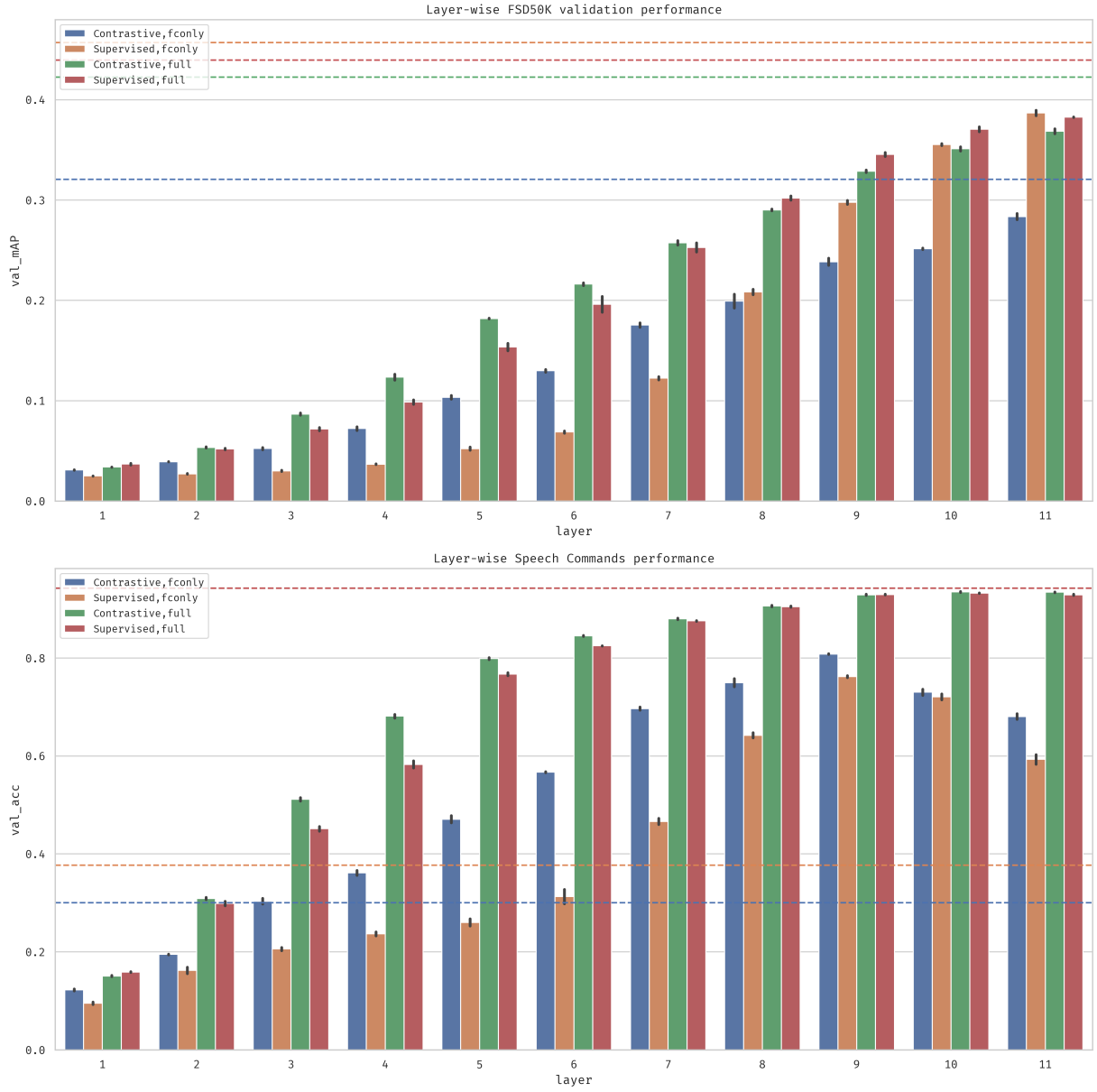

Figure 6.1: Layerwise FSD50K (top) and Speech Commands v2 (bottom) performance for contrastive and supervised pretraining. *mAP* and classification accuracy scores on the validation set are reported, respectively. Dashed lines represent performance without removing any layers. For SpeechCommands, the red dashed line overlaps the green. 95% confidence intervals shown across several runs.

of magnitude smaller than the AudioSet dataset. To this end, we incrementally add layers retained in the final model, followed by temporal average pooling to acclimate for varying feature map sizes in the time-domain, finally adding a classifier on top. Two separate experiments were conducted, the *full* setting where the entire resulting model is fine-tuned and *fconly*, where just the classifier added on top is trained. For FSD50K, *mAP* score is reported on the validation set, whereas classification accuracy on the validation set is reported for Speech Commands v2. Each experiment is repeated three times and 95% confidence intervals are reported over runs.

6.1.3 Results

Figure 6.1 shows steady improvement as we add layers for both contrastive and supervised pretraining in both settings. In general, contrastive pretraining leads to better performance for the initial layers, for both the in-domain and the out-of-domain transfer tasks, more so for the *fconly* setting. However, for the in-domain FSD50K dataset (top), supervised pretraining outperforms contrastive pretraining significantly when deeper layers are added, especially for the *fconly* setting. This demonstrates the importance of class-specific concepts learned during supervised training by the deeper layers, which do a lot of the heavy lifting. For the out-of-domain Speech Commands dataset (bottom), contrastive pretraining performs better than supervised pretraining in all but the final two layers. Interestingly, linear evaluation (linear classifier on top of the full feature encoder) performs worse than adding a linear classifier on top of middle of the stack, suggesting that the frozen feature representations learned in both cases were too specific to sound event recognition.

6.2 Residual signal study for verifying feature map stimulation by sonifications

6.2.1 Objective

In the previous chapter, we discussed how sonifications corresponding to a feature map represented substructures in the input signal that maximally activated the said feature map. The objective of this experiment is to confirm that that's indeed the case, as well as to establish the trend across the various layers of the feature encoder.

6.2.2 Methodology

Given how sonifications correspond to maximally activating input patterns, removing these patterns from the input signal should reduce recognition performance. This is exactly what we do: we analyze how removing these maximally activating input patterns from the input signal affects FSD50K evaluation performance.

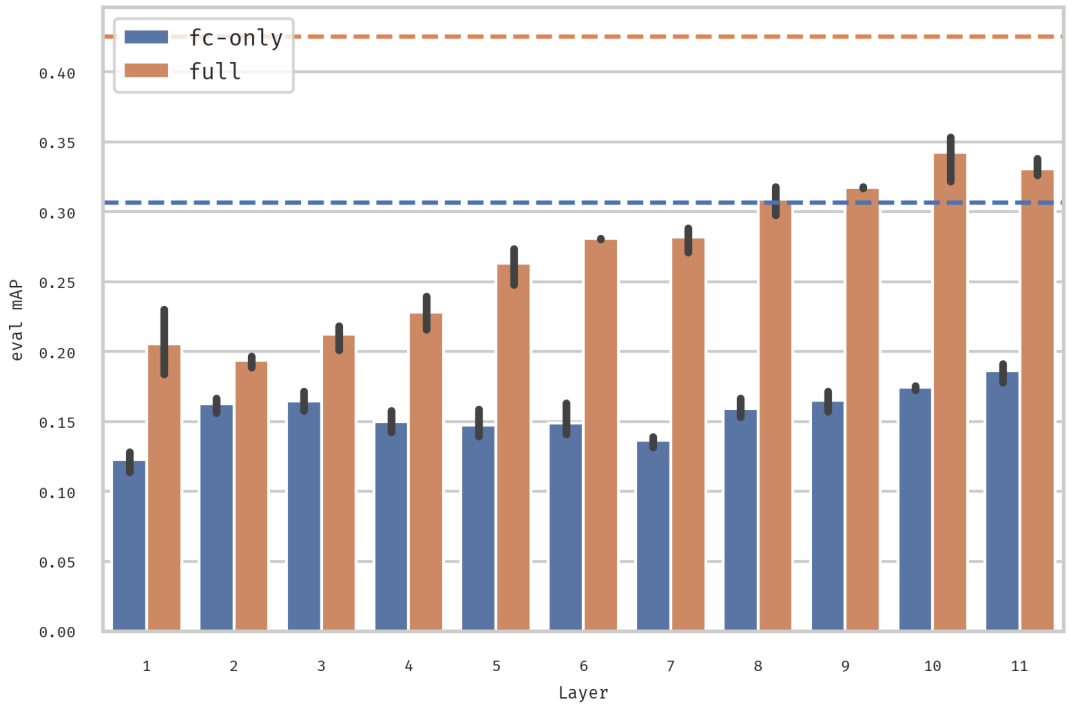

Figure 6.2: Effect of removing sonifications on a layer-by-layer basis from the input on FSD50K eval performance. Contrastive model pretrained on AudioSet was finetuned. Dashed lines represent baseline performance. Removing sonifications from the input signal drastically reduces model performance, indicating that they represent acoustic elements essential for recognition. 95% confidence intervals are computed over several runs.

For every input signal $\boldsymbol{x} \in \mathbb{R}^T$ in the FSD50K evaluation set, we subtract the sonification signal $\boldsymbol{s} \in \mathbb{R}^T$ corresponding to the most active feature map in any given layer. This gives as a *residual input signal* $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{s}$ that represents the input signal stripped of the important audio segments. This is possible since sonifications and their corresponding input signals are time-invariant (Section 5.2.1, Figure 5.1), and thus every timestep t in sonification signal $\boldsymbol{s}$ corresponds to the same timestep in the input signal $\boldsymbol{x}$. Forward pass is executed on the *residual input signal* $\boldsymbol{r}$, and the mean average precision (*mAP*) score is calculated over the evaluation set. The experiment is then repeated for all layers.

6.2.3 Results

Figure 6.2 shows the results of the residual signal study conducted under two settings with contrastive pretrained weights: when the entire model is fine-tuned (*full*) and when only the classifier layer on top is trained (*fconly*). It's evident that sonifications represent elements crucial to recognition performance, as *mAP* scores are drastically reduced. The variance in performance reduction across layers is lower for *fc-only* models, whereas a fully fine-tuned model is more robust, which is an expected result.

6.3 Quantifying similarity between sonifications and inputs

6.3.1 Objective

Section 5.3.2 discusses how sonifications for the contrastive model are more acoustically intelligible as compared to the supervised model, especially for the deeper layers. Given the nature of deconvolution, one can naturally expect acoustic similarity between inputs and sonifications to decrease with network depth, since the reconstruction process now has to recover more and more information. But can this similarity between inputs and corresponding sonifications be quantified and utilized to gain better insights and explanations into the nature of contrastive and supervised representations and how they differ from each other? Answering that question through spectral analysis is the primary objective of this study.

6.3.2 Magnitude-squared coherence

Magnitude-squared coherence between two discrete-time signals is a measurement of correspondence between the two signals, and is a function of their frequency content. It is indicated by values in the $[0, 1]$ range, with higher value at a given frequency denoting higher coherence.

Specifically, the magnitude-squared coherence between two discrete-time signals $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as

$$C_{xy}(f) = \frac{|P_{xy}(f)|^2}{P_{xx}(f) * P_{yy}(f)}, \tag{6.1}$$

where P_{xy} is the cross-spectral density of $\boldsymbol{x}$ and $\boldsymbol{y}$, and P_{xx} and P_{yy} are spectral densities of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively (Stoica & Moses, 2005).

We use *scipy's* (Virtanen et al., 2020) implementation to measure magnitude-squared coherence, which utilizes Welch's method (Welch, 1967) for spectral density estimation. Welch's method is non-parametric in nature, and estimates power spectral density by dividing the signal into overlapping segments in the time domain, followed by averaging a variant of periodogram obtained from each segment.

Mean Top-K magnitude squared coherence: a measure of similarity between inputs and sonifications

To measure how similar the sonifications are to their corresponding inputs for each layer of the feature encoder, the mean magnitude-squared coherence of the top-5 most coherent frequency components between the input signal $\boldsymbol{x}_i$ and the corresponding sonification $\boldsymbol{s}_i$ of the most active feature map of the layer, over the AudioSet evaluation set, is computed. We call this measure mean top-k magnitude squared coherence:

$$M_c = \frac{1}{kN} \sum_{i=1}^{N} \sum \text{top}_{k=5}(C_{x_i s_i}(f)), \tag{6.2}$$

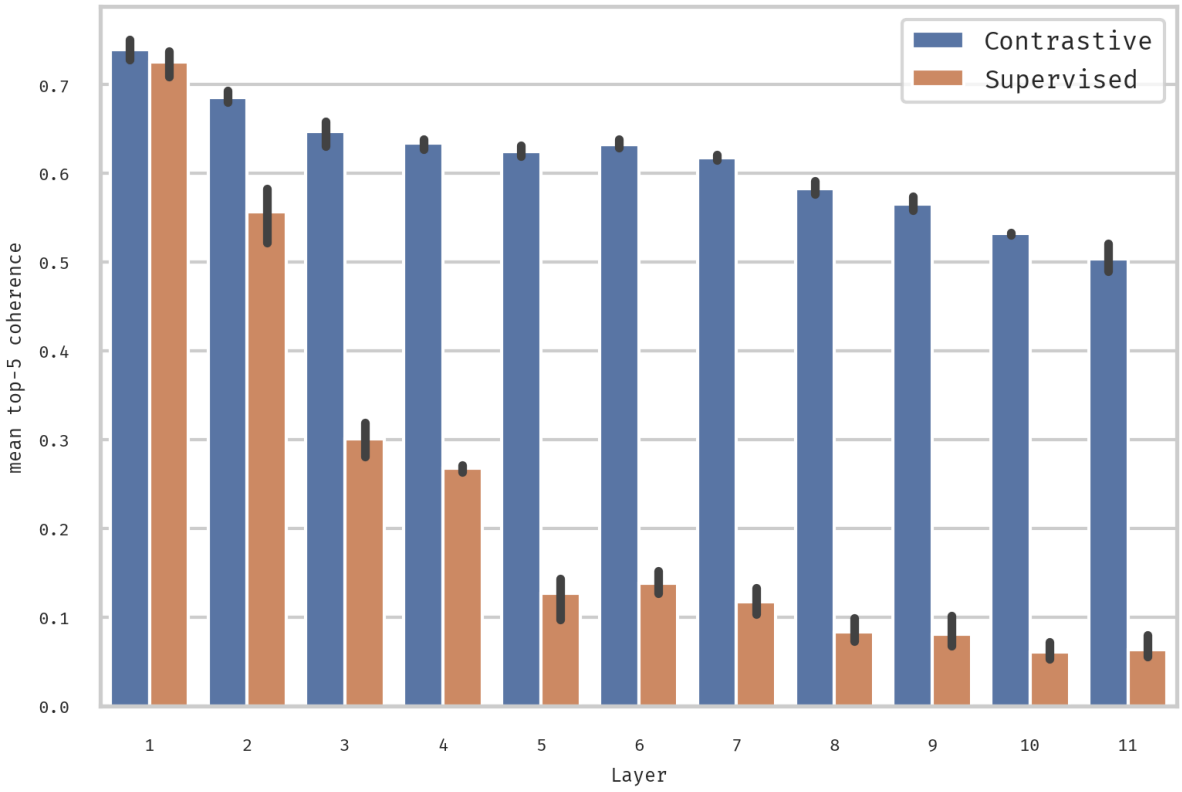

Figure 6.3: Comparing inputs and sonifications of the best contrastive and supervised models trained on AudioSet using magnitude squared coherence, with 95% confidence intervals over three runs.

where N is the number of samples in the AudioSet evaluation set and $\text{top}_k()$ returns the top k values in the input. This is repeated on a layer-by-layer basis for both the contrastive and the supervised (with no pretraining) representations trained on the AudioSet dataset.

6.3.3 Results

Figure 6.3 shows the results of the experiment mentioned above. It is evident that while the measure decreases with layer depth for both the contrastive and the supervised models in line with general consensus, M_c value decreases much more gradually for the contrastive model with layer depth while it decays significantly faster for the supervised model, indicating that the contrastive representations yield sonifications that better correspond with substructures in the input, especially for the deeper layers. This is consistent and supports the observations made in Section 5.3.2, highlighting a significant, quantifiable difference between contrastive self-supervised and supervised representations.

Chapter 7

Conclusion

This work proposed sonification, a method to interpret intermediate feature representations of raw waveform CNNs for sound event recognition by mapping them back into the discrete-time input signal space, where they can be interpreted as intelligible acoustic events.

To this end, transposed convolutions, also known as DeconvNets, were revisited. Extensive performance evaluation was carried out on several benchmark datasets to ensure that the proposed feature encoder, *CNN12*, was fit for analysis (Tables 4.1-4.9).

Using the proposed approachs, supervised and contrastive self-supervised representations based on the SimCLR framework were compared. Inspecting the obtained sonifications, by both listening to them as well as visually inspecting them, highlighted some noteworthy differences between the two training paradigms, viz.

- How deep contrastive layers show greater feature activity while supervised layers are more selective;
- How sonifications from contrastive representations comprise more acoustically perceptible events and richer acoustic content, especially in the deep layers; and
- How contrastive representations show greater acoustic content coupling in the deep layers, while supervised representations focus more on capturing discriminative cues that span a wider variety of maximally activating inputs

The utility of the obtained sonifications to visualize feature evolution while training is demonstrated (Figure 5.5), and an exemplary proof of time-invariance between the sonification signal and the corresponding input is provided in Figure 5.1.

This was followed by several supporting empirical experiments. A layer-wise analysis of transfer learning performance on the FSD50K and the Speech Commands v2 datasets was conducted, verifying that the deep supervised layers contributed to recognition performance despite the lower feature map activity. This was followed by a residual signal study, which verified that sonifications represented input segments crucial for recognition performance. Finally, a spectral

coherence based measure of similarity between the inputs and the corresponding sonifications was proposed, showing how coherence between deep contrastive representations is maintained, supporting our observations regarding the qualitative aspects of the representations from the perspective of acoustic content.

As evidenced by the observations made by listening to the sonifications as well as the outcomes of the empirical experiments, the proposed approach can be used for interpreting intermediate feature representations as legible acoustic events in the input space, possibly aiding debugging of such models and explanability of the decisions made, and the applicability of sonification as a method of interpretation looks encouraging. However, the proposed approach is not without its demerits, and the primary shortcoming is the reliance on the MaxUnpooling operation. MaxUnpooling operation is crucial for maintaining the temporal structure when inverting an intermediate feature map, but limits the approach's applicability to other convolutional neural network architectures. Another shortcoming of the approach is the inability to capture cumulative activity in an entire layer as the deconvolution process only inverts a single feature map at a time, a drawback also discussed by Zeiler & Fergus (2014).

Thus, our next steps consist of alleviating these shortcomings and searching for a more general approach compatible with different DNN architectures and other audio and speech processing tasks, such as automatic speech recognition.

Appendix A

More sonification visualizations

Figures A.1 and A.2 visualize the top-3 activations over the evaluation set of a random set of feature maps (rows) with the inputs signals (left 3x3 grid) and the corresponding sonifications (right 3x3 grid), for each layer in the contrastive and supervised models trained on AudioSet, respectively. Note the lower activity in the deeper layers of the supervised model as compared to the contrastive model.

Figures A.4 and A.3 visualize how features evolve during training. Sonifications for the early layers show the process of learning of band-selective filters. For the deep layers (8-11), the sonification at the first milestone shows a full spectrogram (Fig A.4) and shows how selective abstract cues are learned during training, some becoming dead units.

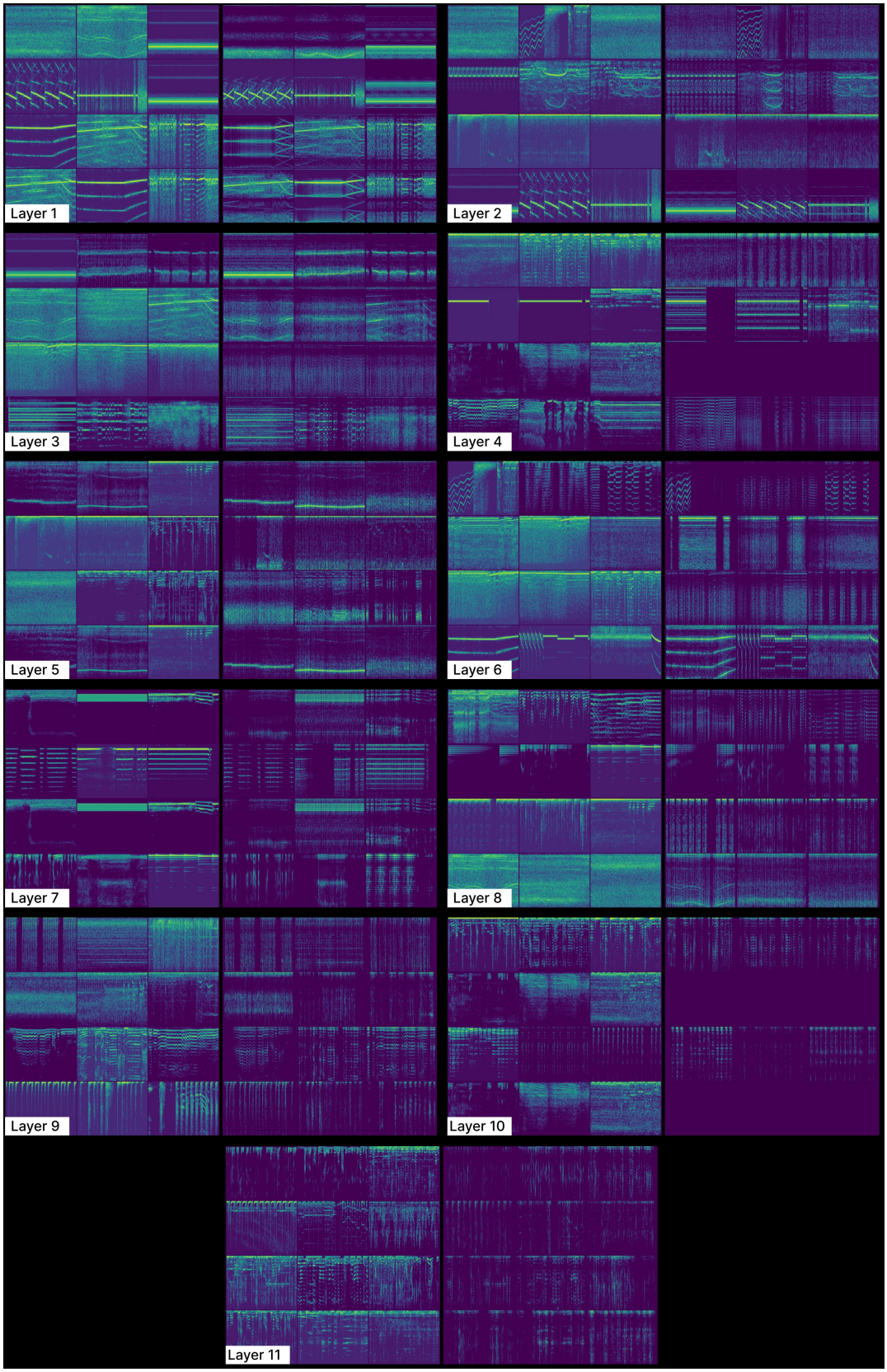

Figure A.1: Sonifications for the contrastive model trained on AudioSet. Every row consists of log-scaled spectrograms of the input and the sonification for the top-3 activations of a random feature map. Best viewed in electronic format.

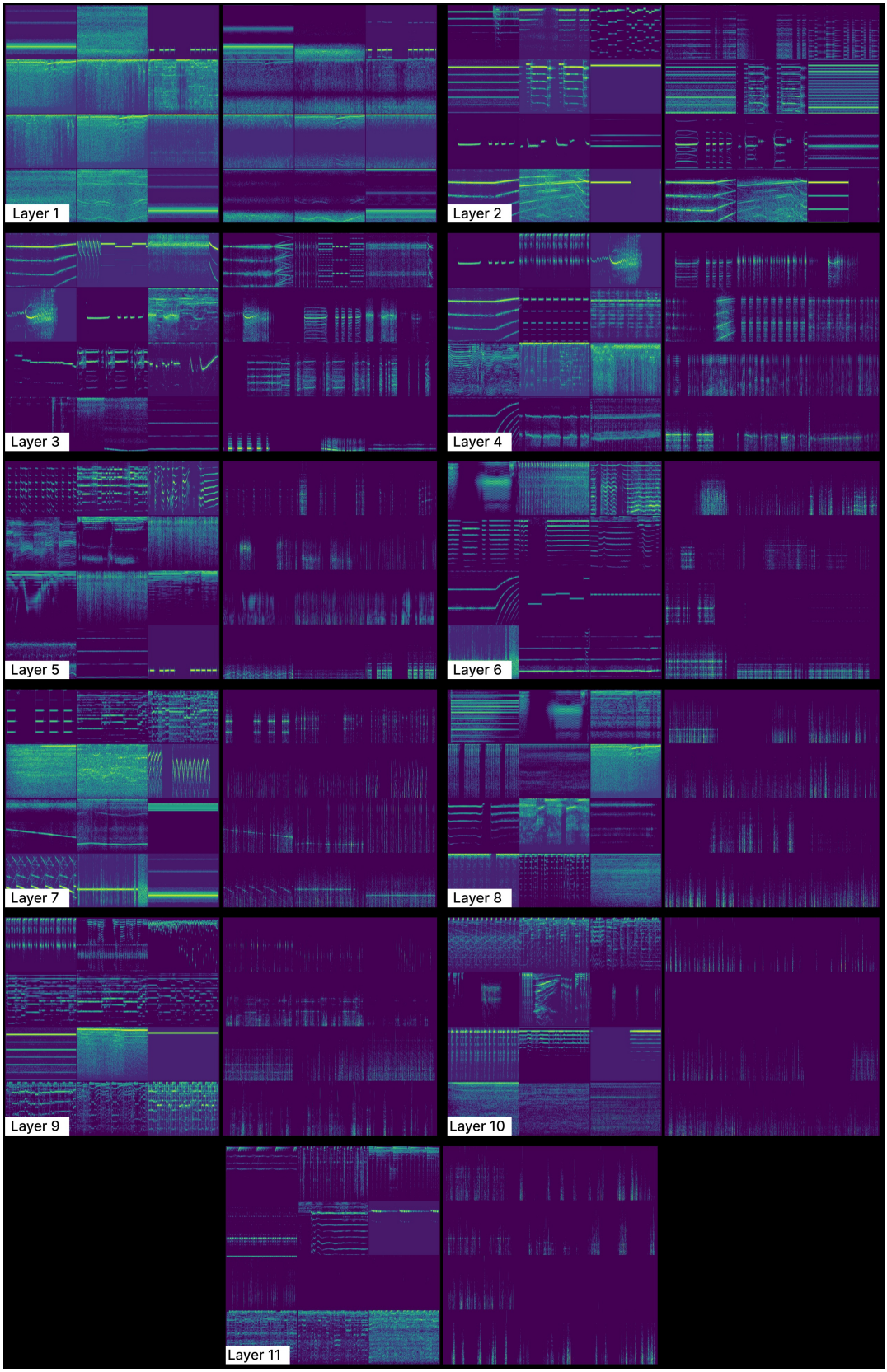

Figure A.2: Sonifications for the supervised model trained on AudioSet, in a similar layout as Figure A.1. Best viewed in electronic format

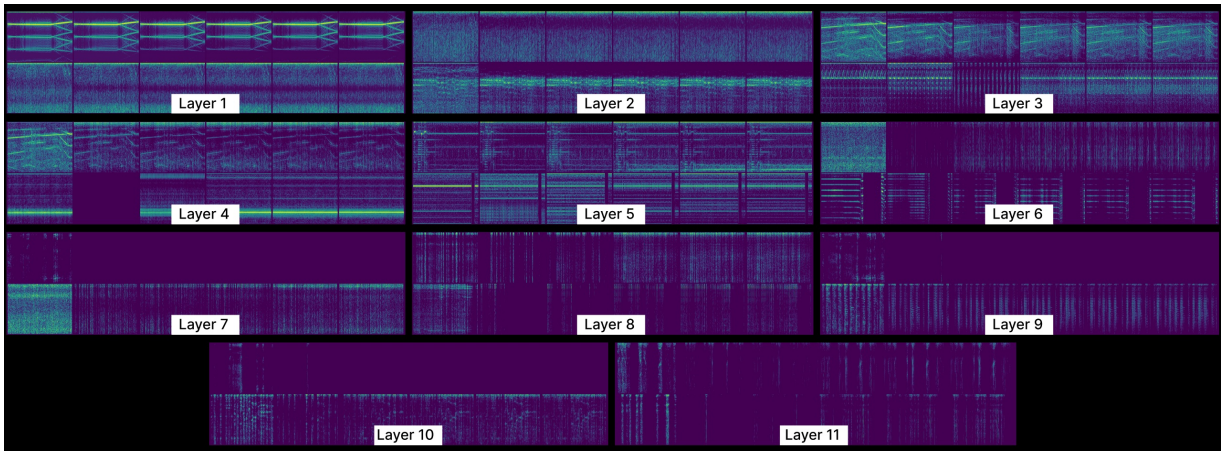

Figure A.3: Contrastive Feature evolution

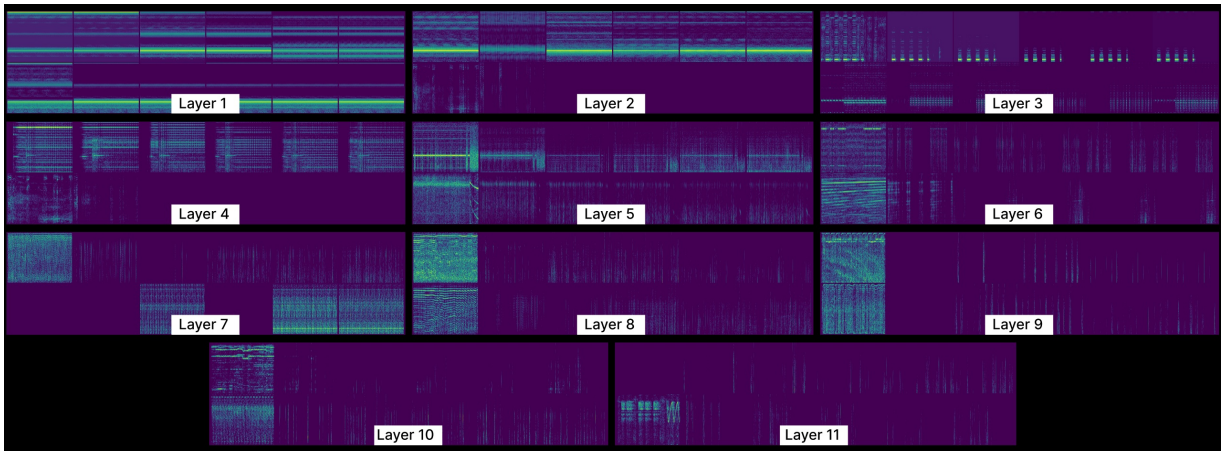

Figure A.4: Visualizing sonifications of random feature maps (rows) for the AudioSet Supervised model at different points in training.

Bibliography

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Vaino Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2: end-to-end speech recognition in English and mandarin. In *ICML 2016: 33rd International Conference on Machine Learning*, pp. 173–182, 2016.

Alexei Baevski and Abdelrahman Mohamed. Effectiveness of Self-Supervised Pre-Training for ASR. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7694–7698, 2020.

Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations. In *ICLR 2020 : International Conference on Learning Representations 2020*, 2020a.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 12449–12460, 2020b.

Gasper Begus and Alan Zhou. Interpreting intermediate convolutional layers of CNNs trained on raw speech. *arXiv preprint arXiv:2104.09489*, 2021.

Gašper Beguš. CiwGAN and fiwGAN: Encoding information in acoustic data to model lexical learning with Generative Adversarial Networks. *Neural Networks*, 139:305–325, 2021.

Lukas Biewald. Experiment Tracking with Weights and Biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

Judith C Brown. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

Judith C Brown and Miller S Puckette. An efficient algorithm for the calculation of a constant Q transform. *The Journal of the Acoustical Society of America*, 92(5):2698–2701, 1992.

Hangting Chen, Zuozhen Liu, Zongming Liu, Pengyuan Zhang, and Yonghong Yan. Integrating the Data Augmentation Scheme with Various Classifiers for Acoustic Scene Modeling. *arXiv preprint arXiv:1907.06639*, 2019a.

Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A Large-Scale Audio-Visual Dataset. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 721–725, 2020a.

Jiaxu Chen, Jing Hao, Kai Chen, Di Xie, Shicai Yang, and Shiliang Pu. An End-to-End Audio Classification System based on Raw Waveforms and Mix-Training Strategy. In *Interspeech 2019*, pp. 3644–3648, 2019b.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML 2020: 37th International Conference on Machine Learning*, volume 1, pp. 1597–1607, 2020b.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Everest Hinton. Big Self-Supervised Models are Strong Semi-Supervised Learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22243–22255, 2020c.

Po-Han Chi, Pei-Hung Chung, Tsung-Han Wu, Chun-Cheng Hsieh, Yen-Hao Chen, Shang-Wen Li, and Hung yi Lee. Audio Albert: A Lite Bert for Self-Supervised Learning of Audio Representation. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pp. 344–350, 2021.

Keunwoo Choi, George Fazekas, Mark Sandler, and Jeonghee Kim. Auralisation of deep convolutional neural networks: Listening to learned features. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, pp. 26–30, 2015.

Keunwoo Choi, George Fazekas, and Mark Sandler. Explaining deep convolutional neural networks on music classification. *arXiv preprint arXiv:1607.02444*, 2016.

Anurag Chowdhury and Arun Ross. DeepVOX: Discovering Features from Raw Audio for Speaker Recognition in Degraded Audio Signals. *arXiv preprint arXiv:2008.11668*, 2020.

Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. VoxCeleb2: Deep Speaker Recognition. In *Interspeech 2018*, pp. 1086–1090, 2018.

Yu-An Chung and James Glass. Generative Pre-Training for Speech with Autoregressive Predictive Coding. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3497–3501, 2020.

Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James R. Glass. An Unsupervised Autoregressive Model for Speech Representation Learning. In *Interspeech 2019*, pp. 146–150, 2019.

Yu-An Chung, Hao Tang, and James R. Glass. Vector-Quantized Autoregressive Predictive Coding. In *Interspeech 2020*, pp. 3760–3764, 2020.

Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2Letter: an End-to-End ConvNet-based Speech Recognition System. *arXiv: Learning*, 2016.

Lionel Delphin-Poulat and Cyril Plapous. MEAN TEACHER WITH DATA AUGMENTATION FOR DCASE 2019 TASK 4. Technical report, Orange Labs Lannion, France, June 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. Ieee, 2009.

Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. ECAPA-TDNN : Emphasized Channel Attention, Propagation and Aggregation in TDNN based speaker verification. In *Interspeech 2020*, pp. 3830–3834, 2020.

Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial Audio Synthesis. In *ICLR 2019 : International Conference on Learning Representations 2019*, 2019.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial Feature Learning. In *ICLR 2016 : International Conference on Learning Representations 2016*, 2016.

Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1538–1546, 2015.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical Report 3, University of Montreal, 2009.

Eduardo Fonseca, Jordi Pons Puig, Xavier Favory, Frederic Font Corbera, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. Freesound datasets: a platform for the creation of open audio datasets. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR*, 2017.

Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel PW Ellis, Xavier Favory, Jordi Pons, and Xavier Serra. General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline. *arXiv preprint arXiv:1807.09902*, 2018.

Eduardo Fonseca, Manoj Plakal, Daniel PW Ellis, Frederic Font, Xavier Favory, and Xavier Serra. Learning sound event classifiers from web audio with noisy labels. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 21–25. IEEE, 2019.

Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. FSD50K: an Open Dataset of Human-Labeled Sound Events. *arXiv preprint arXiv:2010.00475*, 2020.

Eduardo Fonseca, Diego Ortego, Kevin McGuinness, Noel E. O'Connor, and Xavier Serra. Unsupervised Contrastive Learning of Sound Event Representations. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 371–375, 2021.

John S Garofolo. Timit acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*, 1993.

Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780, 2017.

Muriel Gevrey, Ioannis Dimopoulos, and Sovan Lek. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling*, 160(3):249–264, 2003.

Pavel Golik, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. Convolutional Neural Networks for Acoustic Modeling of Raw Time Signal in LVCSR. In *Interspeech 2015*, pp. 26–30, 2015.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of The ACM*, 63(11):139–144, 2020.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML 2006: 23rd International Conference on Machine Learning*, pp. 369–376, 2006.

D. Griffin and Jae Lim. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap Your

Own Latent: A New Approach to Self-Supervised Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284, 2020.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Interspeech 2020*, pp. 5036–5040, 2020.

Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep Speech: Scaling up end-to-end speech recognition. *arXiv: Computation and Language*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9729–9738, 2020.

Hynek Hermansky and Petr Fousek. Multi-resolution RASTA filtering for TANDEM-based ASR. In *Interspeech 2005*, pp. 361–364, 2005.

Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135, 2017.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *IEEE Transactions on Audio, Speech, and Language Processing*, pp. 1–1, 2021a.

Wei-Ning Hsu, Anuroop Sriram, Alexei Baevski, Tatiana Likhomanenko, Qiantong Xu, Vineel Pratap, Jacob Kahn, Ann Lee, Ronan Collobert, Gabriel Synnaeve, and Michael Auli. Robust wav2vec 2.0: Analyzing Domain Shift in Self-Supervised Pre-Training. In *Interspeech 2021*, 2021b.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1857–1867, 2020.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely Connected Convolutional Networks. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Jonathan Huang, Hong Lu, Paulo Lopez Meyer, Hector Cordourier, and Juan Del Hoyo Ontiveros. Acoustic Scene Classification Using Deep Learning-based Ensemble Averaging. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019.

Jui-Ting Huang, Jinyu Li, and Yifan Gong. An analysis of convolutional neural networks for speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4989–4993, 2015.

Seo Hyeji and Park Jihwan. Acoustic Scene Classification Using Various Pre-Processed Features and Convolutional Neural Networks. Technical report, DCASE2019 Challenge, June 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML 2015: 32nd International Conference on Machine Learning*, pp. 448–456. PMLR, 2015.

Selen Hande Kabil, Hannah Muckenhirn, and Mathew Magimai.-Doss. On Learning to Identify Genders from Raw Speech Signal Using CNNs. In *Interspeech 2018*, pp. 287–291, 2018.

Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven Siegelbaum, A James Hudspeth, and Sarah Mack. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

Slawomir Kapka and Mateusz Lewandowski. Sound Source Detection, Localization and Classification using Consecutive Ensemble of CRNN Models. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019.

Kazuya Kawakami, Luyu Wang, Chris Dyer, Phil Blunsom, and Aaron van den Oord. Learning Robust and Multilingual Speech Representations. In *The 2020 Conference On Empirical Methods In Natural Language Processing*, pp. 1182–1192, 2020.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. In *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 28:2880–2894, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25: 1097–1105, 2012.

Andreas Krug and Sebastian Stober. Introspection for convolutional automatic speech recognition. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 187–199, 2018.

Andreas Krug and Sebastian Stober. Visualizing Deep Neural Networks for Speech Recognition with Learned Topographic Filter Maps. *arXiv preprint arXiv:1912.04067*, 2019.

Yann LeCun. Self-supervised learning: could machines learn like humans?, 2018. URL `https://www.youtube.com/watch?v=7I0Qt7GALVk`.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Chung-Yi Li, Pei-Chieh Yuan, and Hung-Yi Lee. What does a network layer hear? analyzing hidden representations of end-to-end asr through speech synthesis. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6434–6438. IEEE, 2020.

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An End-to-End Convolutional Neural Acoustic Model. In *Interspeech 2019*, pp. 71–75, 2019.

Liwei Lin, Xiangdong Wang, Hong Liu, and Yueliang Qian. Guided Learning Convolution System for DCASE 2019 Task 4. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019.

Shaoshi Ling, Yuzong Liu, Julian Salazar, and Katrin Kirchhoff. Deep Contextualized Acoustic Representations for Semi-Supervised Speech Recognition. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6429–6433, 2020.

Alexander H. Liu, Yu-An Chung, and James R. Glass. Non-Autoregressive Predictive Coding for Learning Speech Representations from Local Dependencies. In *Interspeech 2021*, 2021a.

Andy T. Liu, Shang-Wen Li, and Hung yi Lee. TERA: Self-Supervised Learning of Transformer Encoder Representation for Speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 29:2351–2366, 2021b.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR 2016 : International Conference on Learning Representations 2016*, 2016.

David G Lowe. Object recognition from local scale-invariant features. In *1999 IEEE International Conference on Computer Vision (ICCV)*, volume 2, pp. 1150–1157. Ieee, 1999.

Erfan Loweimi, Peter Bell, and Steve Renals. On Learning Interpretable CNNs with Parametric Modulated Kernel-Based Filters. In *Interspeech 2019*, pp. 3480–3484, 2019.

Richard G Lyons. *Understanding digital signal processing, 3/E*. Pearson Education India, 2004.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5188–5196, 2015.

Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A*, 374(2065):20150203–20150203, 2016.

Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. TUT database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pp. 1128–1132. IEEE, 2016.

Hannah Muckenhirn, Vinayak Abrol, Mathew Magimai.-Doss, and Sébastien Marcel. Gradient-based spectral visualization of CNNs using raw waveforms. *Idiap*, 2018a.

Hannah Muckenhirn, Mathew Magimai.-Doss, and Sébastien Marcel. On Learning Vocal Tract System Related Speaker Discriminative Information from Raw Signal Using CNNs. In *Interspeech 2018*, pp. 1116–1120, 2018b.

Hannah Muckenhirn, Mathew Magimai.-Doss, and Sebastien Marcell. Towards Directly Modeling Raw Speech Signal for Speaker Verification Using CNNS. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4884–4888, 2018c.

Hannah Muckenhirn, Vinayak Abrol, Mathew Magimai-Doss, and Sébastien Marcel. Understanding and Visualizing Raw Waveform-based CNNs. In *Interspeech 2019*, pp. 2345–2349, 2019.

Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. VoxCeleb: A Large-Scale Speaker Identification Dataset. In *Interspeech 2017*, pp. 2616–2620, 2017.

Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. BYOL for Audio: Self-Supervised Learning for General-Purpose Audio Representation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021.

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning Deconvolution Network for Semantic Segmentation. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1520–1528, 2015.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Dimitri Palaz, Ronan Collobert, and Mathew Magimai.-Doss. Estimating Phoneme Class Conditional Probabilities from Raw Speech Signal using Convolutional Neural Networks. In *Interspeech 2013*, pp. 1766–1770, 2013.

Dimitri Palaz, Mathew Magimai.-Doss, and Ronan Collobert. ANALYSIS OF CNN-BASED SPEECH RECOGNITION SYSTEM USING RAW SPEECH AS INPUT. In *Interspeech 2015*, pp. 11–15, 2015.

Dimitri Palaz, Mathew Magimai-Doss, and Ronan Collobert. End-to-end acoustic modeling using convolutional neural networks for HMM-based automatic speech recognition. *Speech Communication*, 108:15–32, 2019.

Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *Journal of Machine Learning Research*, 18(1):2887–2938, 2017.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin Dogus Cubuk, and Quoc V. Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Interspeech 2019*, pp. 2613–2617, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8026–8037, 2019.

Nathanael Perraudin, Peter Balazs, and Peter L. Sondergaard. A fast Griffin-Lim algorithm. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 1–4, 2013.

Karol J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018. ACM Press, 2015. ISBN 978-1-4503-3459-4. doi: 10.1145/2733373.2806390. URL `http://dl.acm.org/citation.cfm?doid=2733373.2806390`.

Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge J. Belongie, and Yin Cui. Spatiotemporal Contrastive Video Representation Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6964–6974, 2020.

Mirco Ravanelli and Yoshua Bengio. Interpretable Convolutional Filters with SincNet. *arXiv: Audio and Speech Processing*, 2018a.

Mirco Ravanelli and Yoshua Bengio. Speaker Recognition from Raw Waveform with SincNet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 1021–1028, 2018b.

Mirco Ravanelli and Yoshua Bengio. Speech and Speaker Recognition from Raw Waveform with SincNet. *arXiv: Audio and Speech Processing*, 2019.

Morgane Riviere, Armand Joulin, Pierre-Emmanuel Mazare, and Emmanuel Dupoux. Unsupervised Pretraining Transfers Well Across Languages. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7414–7418, 2020.

Aaqib Saeed, David Grangier, and Neil Zeghidour. Contrastive Learning of General-Purpose Audio Representations. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3875–3879, 2021.

Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A Dataset and Taxonomy for Urban Sound Research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 1041–1044, 2014.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised Pre-Training for Speech Recognition. In *Interspeech 2019*, pp. 3465–3469, 2019.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, 128(2):336–359, 2020.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *ICLR 2013 : International Conference on Learning Representations 2013 (Workshop)*, 2013.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Steven W. Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Pub. San Diego, 1997.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-Vectors: Robust DNN Embeddings for Speaker Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5329–5333, 2018.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pp. 1857–1865, 2016.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for Simplicity: The All Convolutional Net. In *ICLR 2015 : International Conference on Learning Representations 2015 (Workshop)*, 2015.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway Networks. *arXiv: Learning*, 2015.

Robert J Sternberg, Karin Sternberg, and Jeff Mio. *Cognitive psychology*. Cengage Learning Press, 2012.

P. G. Stoica and Randolph L. Moses. *Spectral analysis of signals*. Prentice Hall, 2005.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive Multiview Coding. In *16th European Conference on Computer Vision, ECCV 2020*, pp. 776–794, 2020.

Nicolas Turpault, Romain Serizel, Ankit Parag Shah, and Justin Salamon. Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Prateek Verma and Jonathan Berger. Audio Transformers: Transformer Architectures For Large Scale Audio Understanding. Adieu Convolutions. *arXiv: Sound*, 2021.

Prateek Verma and Ronald W. Schafer. Frequency Estimation from Waveforms Using Multi-Layered Neural Networks. In *Interspeech 2016*, pp. 2165–2169, 2016.

Jesse Vig. A Multiscale Visualization of Attention in the Transformer Model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 37–42, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3007. URL `https://www.aclweb.org/anthology/P19-3007`.

Jesse Vig and Yonatan Belinkov. Analyzing the Structure of Attention in a Transformer Language Model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 63–76, 2019.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J van der Walt, Matthew Brett, Joshua Wilson, K Jarrod Millman, Nikolay Mayorov, Andrew R J Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W Moore,

Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E A Quintero, Charles R Harris, Anne M Archibald, Antônio H Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy . Contributors. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

Luyu Wang, Pauline Luc, Yan Wu, Adria Recasens, Lucas Smaira, Andrew Brock, Andrew Jaegle, Jean-Baptiste Alayrac, Sander Dieleman, Joao Carreira, et al. Towards Learning Universal Audio Representations. *arXiv preprint arXiv:2111.12124*, 2021.

Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv preprint arXiv:1804.03209*, 2018.

P. Welch. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.

Jee weon Jung, Hye jin Shim, Ju ho Kim, Seung bin Kim, and Ha-Jin Yu. Acoustic Scene Classification using Audio Tagging. In *Interspeech 2020*, pp. 1176–1180, 2020.

Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *15th European Conference on Computer Vision, ECCV 2018*, pp. 3–19, 2018.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3733–3742, 2018.

Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Utterance-level Aggregation for Speaker Recognition in the Wild. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5791–5795, 2019.

Sarthak Yadav and Mary Ellen Foster. Revisiting transposed convolutions for interpreting raw waveform sound event recognition CNNs by sonification, 2022. URL `https://openreview.net/forum?id=uecYQBshVYV`.

Sarthak Yadav and Atul Rai. Learning Discriminative Features for Speaker Identification and Verification. In *Interspeech 2018*, pp. 2237–2241, 2018.

Sarthak Yadav and Atul Rai. Frequency and Temporal Convolutional Attention for Text-Independent Speaker Recognition. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6794–6798, 2020.

Neil Zeghidour, Olivier Teboul, Félix de Chaumont Quitry, and Marco Tagliasacchi. LEAF: A Learnable Frontend for Audio Classification. In *ICLR 2021 : International Conference on Learning Representations 2021*, 2021.

Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *13th European Conference on Computer Vision, ECCV 2014*, pp. 818–833, 2014.

Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pp. 2018–2025. IEEE, 2011.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. In *ICLR 2017 : International Conference on Learning Representations 2017*, 2017a.

Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 645–654, 2017b.

Zhou and Chellappa. Computation of optical flow using a neural network. In *IEEE 1988 International Conference on Neural Networks*, pp. 71–78, 1988.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent Highway Networks. In *ICML 2017: 34th International Conference on Machine Learning*, pp. 4189–4198, 2017.