REVISITING TRANSPOSED CONVOLUTIONS FOR INTERPRETING RAW WAVEFORM SOUND EVENT RECOGNITION CNNS BY SONIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Most recent work on the interpretability of audio and speech processing deep neural networks (DNNs) interprets spectral information modelled by the first layer, relying solely on visual means of interpretation. In this work, we propose *sonification*, a method to interpret intermediate feature representations of sound event recognition (SER) convolutional neural networks (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. We use sonification to compare supervised and self-supervised feature hierarchies and observe that deep self-supervised layers learn more acoustically discernible representations compared to their supervised counterparts. We also quantify similarity between the sonifications and their corresponding inputs using spectral coherence on a layer-by-layer basis.

1 INTRODUCTION

Interpretability of deep neural networks (DNNs) has always garnered much attention. DNNs learn highly expressive feature hierarchies, and with the ever-growing push towards deeper and larger networks with tighter end-to-end integration that comes at the cost of higher abstraction, together with the rise in popularity of self-supervised representation learning, the importance of understanding the inner workings of DNNs continues to rise.

Visualizations have emerged as the most prominent method of interpreting DNNs, whether it be the visualization of attention mechanisms (Woo et al., 2018; Vaswani et al., 2017; Vig, 2019; Vig & Belinkov, 2019), or interpreting deep feature representations (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), visualizations cross boundaries of input modalities and neural architectures. And rightly so, the visual cortex is our brain's most essential and complex sensory perception system (Kandel et al., 1981; Sternberg et al., 2012), possibly making visualizations more immediate than other forms of interpretation. The audio and speech processing domain has not remained untouched. A significant body of work attempts to interpret spectral information captured by DNN filters and parameters by 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). However, if the pertinent question is ascertaining what DNNs learn and model, would interpreting intermediate feature representations in the audio input space where they can be directly perceived as intelligible acoustic elements be beneficial over *solely* relying on visual means of description? This is the primary motivation behind this paper: to interpret sound event recognition CNNs by sonification, i.e. mapping intermediate feature representations back into discrete-time audio signal input space, and can be seen as an extension of Zeiler & Fergus (2014) for raw waveform audio CNNs. More specifically, in this paper:

- We revisit transposed convolutions (a.k.a. deconvolutions) for interpreting intermediate feature representations of *raw waveform based 1D-CNNs for sound event recognition* (SER) by sonification, highlighting maximally activating substructures as intelligible acoustic events.

- Using sonifications, we compare and contrast raw waveform based supervised and self-supervised feature representations in the discrete-time signal input space.
- Using Spectral Coherence (Stoica & Moses, 2005), we show how sonifications compare to corresponding input signals as we go deeper into the self-supervised and supervised feature hierarchies.

2 RELATED WORK

Several recent works in the audio processing domain emphasize on the interpretability of DNNs, especially convolutional neural networks applied on automatic speech recognition (ASR) and speaker recognition. In ASR, several works visualize filters learned from spectrogram inputs (Huang et al., 2015; Krug & Stober, 2018; 2019). Huang et al. (2015) visualize spectro-temporal filters while Krug & Stober (2019) learn topographic filter maps in the first convolutional layer. The majority 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. By visualizing the magnitude spectrum of the filters sorted by the estimated center frequency, Golik et al. (2015) showed that the majority of the filters of the first convolutional layer have learned narrow bandpass filters. Palaz et al. (2019) show that filters in the first convolutional layer model formant related information to learn a phone-discriminant spectral dictionary, a notion that is also supported by Mallat (2016); Papyan et al. (2017); Kabil et al. (2018); Muckenhirn et al. (2018b). For the most part, things are quite similar in the speaker recognition domain. Muckenhirn et al. (2018c;b) visualize and interpret the cumulative frequency response of the first convolutional layer trained on raw waveforms, and show how fundamental frequency and formant information is modelled by the filters. Ravanelli & Bengio (2018); Zeghidour et al. (2021) propose an interpretable raw waveform front-end, visualizing their magnitude frequency response.

Inspired by the guided backpropagation (Springenberg et al., 2015) approach from computer vision, several works (Muckenhirn et al., 2019; 2018a; Chowdhury & Ross, 2020) propose the usage of gradient-based relevance signals to interpret intermediate feature representations of raw waveform models, as they also describe periodicity information in the time domain, whereas Krug & Stober (2018) utilize activation maximisation (Erhan et al., 2009) for global introspection. 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 based on Highway Networks as a probe, trained separately for each layer. Begus & Zhou (2021) interpret intermediate representations learned by a WaveGAN (Donahue et al., 2019) model trained on raw speech data in the time-domain by using a ciwGAN (Beguš, 2021) model as a probe.

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 work that focuses 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. 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. Paired with the attention raw waveform based deep self-supervised audio and speech representations have received recently, there is a dire need for methods that improve interpretability of raw waveform models. To this end, 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. Deconvolutions have been used previously in the computer vision domain for revealing remarkable insights about image recognition CNNs in the foundational work by Zeiler & Fergus (2014). Deconvolutions were also used by Choi et al. (2015; 2016) to interpret intermediate representations of *2D-CNNs trained on spectrogram inputs for music classification*, an approach that allowed them to treat the input space as a pixel representation in the frequency domain, facilitating analysis. Since Choi et al. (2015; 2016) work in the frequency domain they use the inverse short-term fourier transform (inverse STFT) to map representations into the discrete-time input space, as compared to the proposed approach, which analyses raw waveform representations and maps intermediate feature

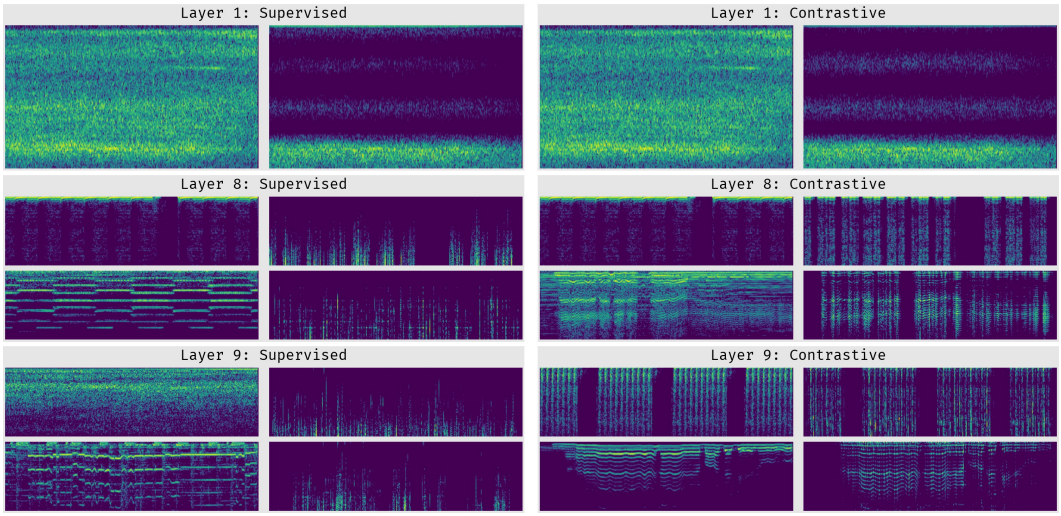

Figure 1: Comparing band selectivity and feature map activity across supervised and contrastive representations by visualising input-sonification pairs as log-scaled spectrograms. Note the greater activity in the higher frequency bins of the contrastive representations for layer 8 and 9.

representations directly into the discrete-time input space. Choi et al. (2016) also do not compare how input signals and their respective interpretations correspond with increasing network depth.

3 SONIFICATION

3.1 PREREQUISITES

To test the proposed method, we set up a simple 12-layer 1D-CNN, named *CNN12* as shown in Figure A.1, as our feature encoder, $f_{enc}(.)$. *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 $\boldsymbol{x} \in \mathbb{R}^T$, $x_i \in \{-1, 1\}$ 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. It's worth noting that the only pre-processing applied on the input to our feature encoder is scaling the input to unit peak amplitude, which significantly simplifies the sonification process

Rather than going for the state-of-the-art, our objective was to establish an easy-to-follow neural architecture which is sufficiently performant, and is not tied to a specific raw waveform front-end (such as Ravanelli & Bengio (2018); Zeghidour et al. (2021)). This is reflected in our design choices: CNNs with similar number of layers are quite common (Simonyan & Zisserman, 2015; Collobert et al., 2016; Kong et al., 2020), and the kernel width and stride hyperparameters were based on known good starting points. We do not use normalization layers like Batch Normalization and Instance Normalization in order to keep the sonification process simple and independent from batch statistics. This also circumvents the need for using globally synchronized normalization techniques in distributed training, which is necessary to avoid local information leakage in contrastive self-supervised models and learn better representations (Chen et al., 2020).

We train contrastive self-supervised feature representations on AudioSet using the SimCLR framework (Chen et al., 2020), which has already been demonstrated to work well in the audio event recognition domain by Fonseca et al. (2021); Saeed et al. (2021). We use an MLP with one-hidden layer with 512 hidden units as our projection head $f_{proj}(.)$ to extract features $\boldsymbol{z} = f_{proj}(f_{enc}(\boldsymbol{x})), \boldsymbol{z} \in \mathbb{R}^{d=512}$, on which *NT-Xent* loss (Chen et al., 2020) is used:

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

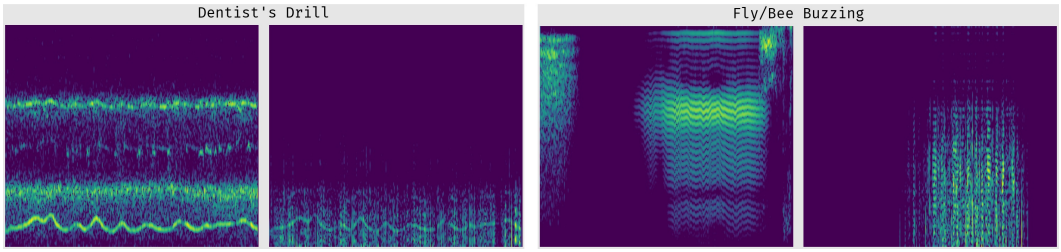

Figure 2: Select supervised input-sonification pairs. For supervised models, most remarkable sonifications were found in the middle layers (layers 3-7).

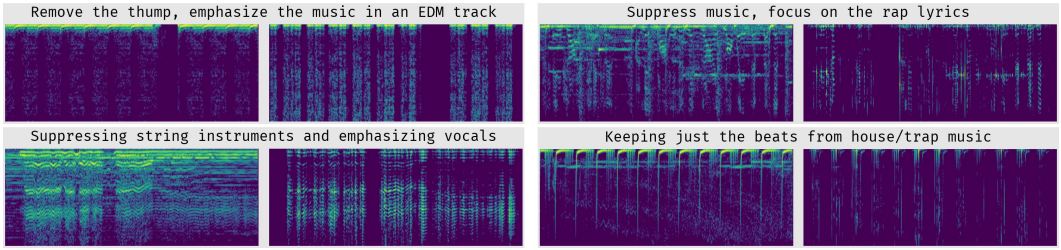

Figure 3: Select contrastive input-sonification pairs. These samples are from layers 8 and 11, and similarly remarkable sonifications can be found throughout all the layers.

where $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ are the representations for the positive, non-overlapping patches $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ that are 2.5 seconds each in duration, $\tau > 0$ 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, and $sim(\boldsymbol{u}, \boldsymbol{v})$ denotes the cosine similarity between two ℓ_2-normalized vectors.

This is followed by an analysis of supervised learning performance on AudioSet, FSD50K (Fonseca et al., 2020) and the Speech Commands v2 (Warden, 2018) datasets to establish that the proposed framework is performant enough for analysis. 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), and all experiments were done on a TPUv3-8 machine. More details, including relevant hyperparameters, data augmentation methods used, training setup, and benchmark results can be found in Appendix A.

3.2 How sonification works

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 B.3). 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 $\boldsymbol{s}$.

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 obtained sonifications are maximally activating patterns in the input signal and not *generated* samples: as demonstrated in Sec 4.1, removing these sonifications from the input signals significantly alters feature map activity resulting in performance degradation. Pseudocode for the sonification process can be found in B.2.

3.3 Observations

This section covers our observations made after inspecting sonifications corresponding to the top-3 maximally activating inputs for each filter in each layer of the *CNN12* feature encoder. We listened to input-sonification pairs, as well as visually inspected them in the frequency domain as log-scaled spectrograms. You can also listen to the sonifications on our anonymous *Weights and Biases* (Biewald, 2020) dashboards for supervised[1] and contrastive[2] representations.

3.3.1 Band Selectivity and Feature Map Activity

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 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 1 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.

3.3.2 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 2). 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.

[1] https://wandb.ai/paper1517iclr2022/select-sonifications-supervised

[2] https://wandb.ai/paper1517iclr2022/select-sonifications-contrastive

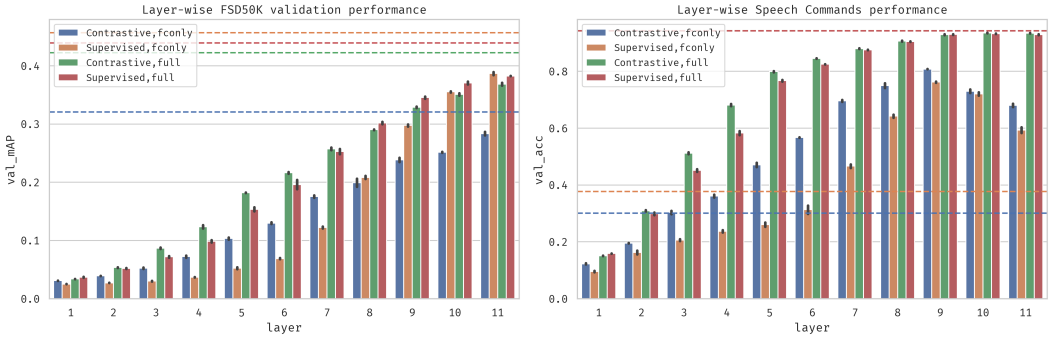

Figure 4: Layerwise FSD50K (left) and Speech Commands v2 (right) performance for contrastive and supervised pretraining. *mAP* and classification accuracy scores on the validation set are reported, respectively. Dashed lines represent performance without removing any layers. For Speech-Commands, the red dashed line overlaps the green. 95% confidence intervals shown across several runs. We conducted this study to rule out suspiciously low feature map activity that we observed when evaluating sonifications for the deep layers of the supervised model and ascertain that these layers were indeed contributing to recognition performance. More information in B

- 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 4), do not correspond to intelligible acoustic events in the inputs, but merely indicate the presence of them, as evidenced by the wider variety of input signals that can maximally activate the feature maps (Springenberg et al., 2015).

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 4.2. 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 3). 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. This is in contrast with supervised representations, where we observed a wider variety of maximally activating input signals, inline with previous observations (Springenberg et al., 2015).

3.4 Using sonifications to visualize training time feature evolution

Sonifications can also be used to visualize how features evolve during training time. We find the maximally activating input for randomly selected feature maps in the AudioSet evaluation set for 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 gives us a smoother visualization of training time feature progression than Zeiler & Fergus (2014), where sudden jumps appeared in the visualization whenever the maximally activating input changed. Figure 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.

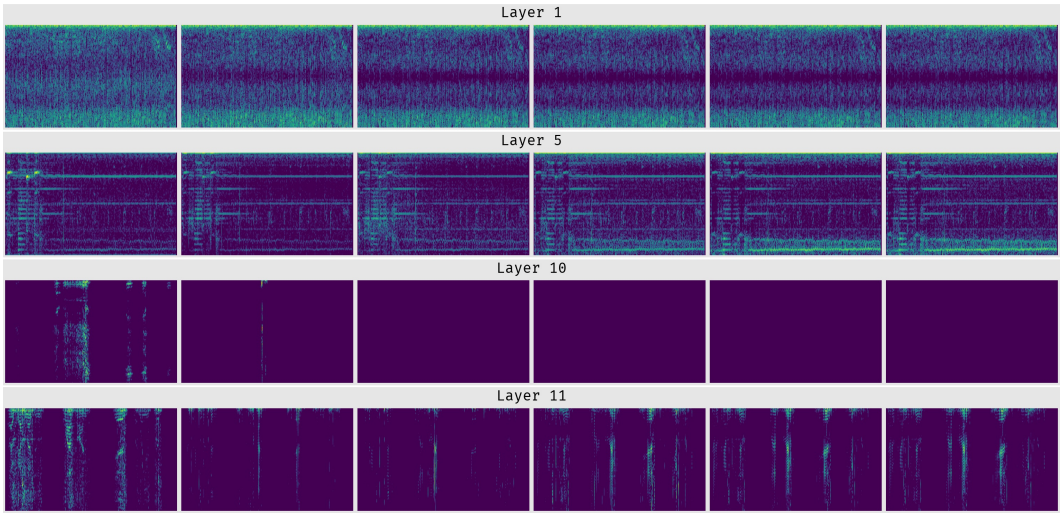

Figure 5: Contrastive feature evolution in the select feature maps from several layers. Note how certain frequency bins are being increasingly emphasised in the layer 5 feature map, while dead feature map in layer 10 starts with a full spectrogram, but fades away later.

4 Experiments and Analysis

4.1 Verifying that sonifications indeed stimulate feature maps

Section 3.2 discussed how sonifications for a feature map correspond to maximally activating sub-structures in the input audio space. We now verify that these sonifications stimulate the feature maps by analyzing how removing them from the input signal affects event recognition performance on the FSD50K evaluation set.

To demonstrate how sonifications stimulate feature maps, for every input signal $\boldsymbol{x} \in \mathbb{R}^T$ in the FSD50K evaluation set, we subtract the sonified signal $\boldsymbol{s} \in \mathbb{R}^T$ corresponding to the most active feature map of the layer from the input signal to yield a *residual input signal* $\boldsymbol{r} \in \mathbb{R}^T$. This is possible since inputs and sonifications are time-invariant, and thus every timestep in the sonification signal corresponds to the same timestep in the input signal. The residual signal $\boldsymbol{r}$ represents the input signal stripped of the audio segments that maximally activate the feature map of interest. We then calculate the mean average precision (*mAP*) score over the evaluation set, comparing it with baseline performance. From Figure 6, 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 finetuned model is more robust.

4.2 Quantifying similarity between inputs and sonifications

In Section 3.3 we observed how sonifications for the contrastive model lead to more intelligible acoustic events as compared to the supervised model, specially for the deeper layers. It raises the question as to how similar the sonifications and their corresponding inputs are, and how do they compare across supervised and contrastive models. Given the nature of deconvolution, one can expect that similarity between inputs and sonifications decreases as we go deeper into the network, since reconstruction becomes more difficult with increasing depth. However, the question is how can this be quantified? To this end we use spectral analysis of the input and the corresponding sonification obtained with respect to a feature map.

4.2.1 Mean top-k magnitude-squared coherence

Magnitude-squared coherence between two discrete-time signals is a function of their frequency content and indicates how well the signals correspond to each other at each frequency, indicated by

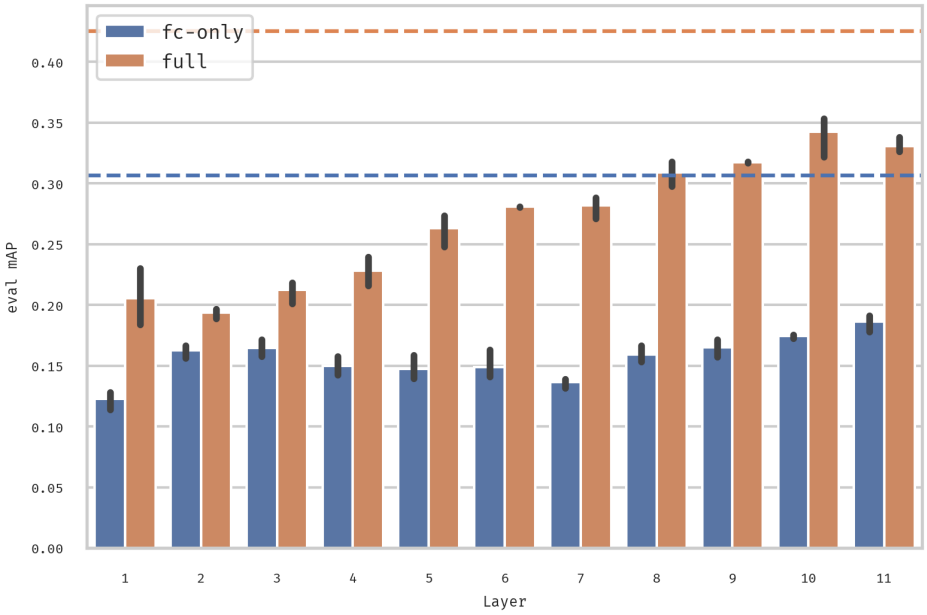

Figure 6: Effect of removing sonifications on a layer-by-layer basis from the input on FSD50K eval performance. Contrastive model pretrained on AudioSet was finetuned. Dashed lines represent baseline performance. Removing sonifications from the input signal drastically reduces model performance, indicating that they represent acoustic elements essential for recognition. 95% confidence intervals are computed over several runs.

values in the $[0, 1]$ range, with higher values showing higher coherence. Specifically, the magnitude-squared coherence between two discrete-time signals $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as

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

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 measure the mean magnitude-squared coherence of the 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, over the AudioSet Evaluation Set, on a layer-by-layer basis (Equation 3).

$$M_c = \frac{1}{kN} \sum_{i=1}^{N} \sum \max_{k} (C_{x_i s_i}(f)), \tag{3}$$

where N is the number of samples in the AudioSet evaluation set, and $k = 5$.

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. Figure 7 shows that while the measure decreases with layer depth for both the contrastive and the supervised models, which was expected, it decays significantly faster for the latter and is consistent with the observations made in Section 3.3, and highlights a significant, quantifiable difference between contrastive and supervised representations.

5 Conclusion

In this paper, we revisited transposed convolutions to interpret intermediate feature representations of raw waveform-based CNNs for sound event recognition by mapping feature representations back into the discrete-time input signal space, resulting in their interpretation as intelligible acoustic

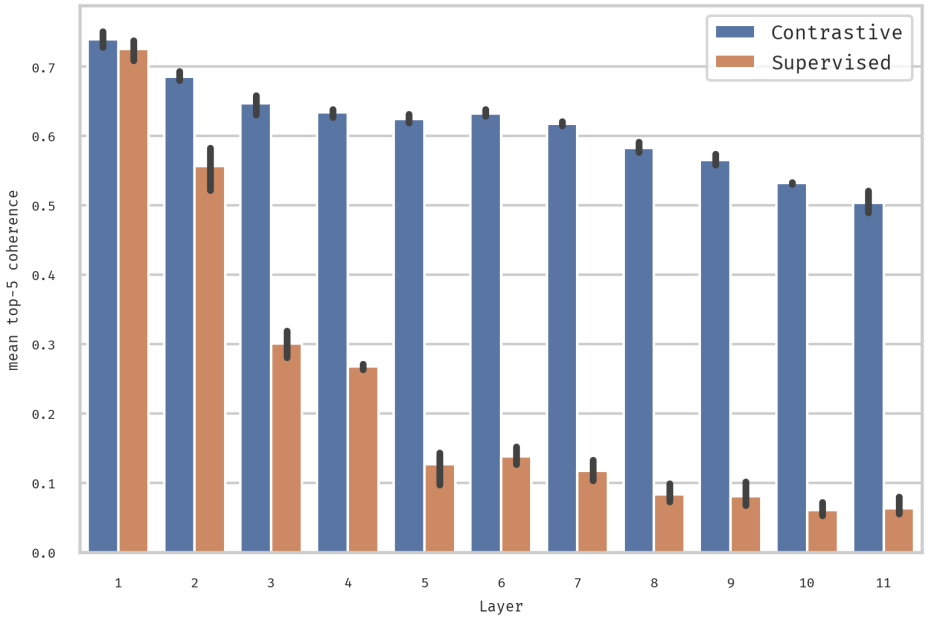

Figure 7: Comparing inputs and sonifications of the best contrastive and supervised models trained on AudioSet using magnitude squared coherence, with 95% confidence intervals over three runs.

events. Using the proposed method, we compare supervised and contrastive self-supervised feature representations and quantify similarity between inputs and corresponding sonifications using spectral coherence, highlighting key differences between the two paradigms. In future work, we will move towards further improving the proposed approach, particularly reducing reliance on the MaxUnpooling operation and adding the ability to measure the cumulative response of the entire layer. We hope that this paper will inspire more work on the interpretability of audio processing DNNs in the time-domain input signal space.

6 Acknowledgments

This research was supported by the *TPU Research Cloud (TRC) program*[3], a Google Research initiative, which gave us access to TPU v2 and v3 devices. We would like to thank the reviewers at ICLR 2022 for their feedback and suggestions that proved invaluable in improving the quality of this work. We would also like to thank AV and PSR for helpful discussions.

7 Reproducibility Statement

To make the experiments reproducible, all the code, including exact hyperparameters will be released under an open source license, which is also mandated by our participation in the TRC program.

[3]https://sites.research.google/trc/about/

REFERENCES

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.

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, 2019.

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, 2020.

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.

Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv: Learning*, 2016.

Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In *ICLR*, 2019.

Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *2015 IEEE 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. *University of Montreal*, 1341(3):1, 2009.

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 *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 371–375, 2021.

Pavel Golik, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. Convolutional neural networks for acoustic modeling of raw time signal in lvcsr. In *16th Annual Conference of the International Speech Communication Association*, pp. 26–30, 2015.

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.

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, and Thomas M. Jessell. *Principles of Neural Science*. 1981.

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.

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.

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 *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6434–6438. IEEE, 2020.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR (Poster)*, 2016.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE 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.

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

Hannah Muckenhirn, Mathew Magimai.-Doss, and Sébastien Marcel. On learning vocal tract system related speaker discriminative information from raw signal using cnns. In *Proceedings of Interspeech*, 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 *Proceedings of Interspeech*, pp. 2345–2349, 2019.

Dimitri Palaz, Ronan Collobert, and Mathew Magimai.-Doss. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. In *Proceedings of Interspeech*, 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 *Proceedings of Interspeech*, 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.

Mirco Ravanelli and Yoshua Bengio. Interpretable convolutional filters with sincnet. *arXiv: Audio and Speech Processing*, 2018.

Aaqib Saeed, David Grangier, and Neil Zeghidour. Contrastive learning of general-purpose audio representations. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3875–3879, 2021.

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 and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR (Workshop Poster)*, 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.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 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*. 2005.

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. *arXiv preprint arXiv:1906.04284*, 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.

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.

Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Neil Zeghidour, Olivier Teboul, Félix de Chaumont Quitry, and Marco Tagliasacchi. Leaf: A learnable frontend for audio classification. In *ICLR 2021: The Ninth International Conference on Learning Representations*, 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 International Conference on Computer Vision*, pp. 2018–2025. IEEE, 2011.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2017.

A SETTING UP THE BASELINE

A.1 1D-CNN FEATURE ENCODER

Figure A.1 shows the CNN12 feature encoder used. We also benchmark an encoder that has 2x the number of filters in each convolutional layer, which we refer to as *CNN12*_x2. All the experiments in the main text are conducted on *CNN12*.

```
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 A.1: CNN12 feature encoder

A.2 CONTRASTIVE PRETRAINING

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 A.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

As mentioned before, a MLP with one-hidden layer with 512 hidden units as our projection head $f_{proj}(.)$ to extract features $\boldsymbol{z} = f_{proj}(f_{enc}(\boldsymbol{x})), \boldsymbol{z} \in \mathbb{R}^{d=512}$, on which *NT-Xent* loss (Chen et al., 2020) was used (Eq 1).

Additive Gaussian noise, random timestep masking, random gain augmentation and finally, background noise injection preserving random signal-to-noise ratio (SNR), which is similar to *mix-back* augmentation utilized by Fonseca et al. (2021), are used (Table A.1). Linear evaluation, i.e. training a classifier on top of the features extracted from a "frozen" feature encoder, was used for quantifying contrastive model performance. Hyperparameter tuning was done on a random holdout set comprising 10% of the "unbalanced train" samples. Adam optimizer 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.

A.3 SUPERVISED LEARNING EXPERIMENTS

For supervised learning, we add a fully connected layer on top of f_{enc}, trained by solving a classification loss, which is either multilabel binary cross-entropy (for AudioSet, FSD50K) or multiclass categorical cross-entropy (Speech Commands v2). Apart from experiments that require training the entire model on the entire AudioSet dataset, all supervised learning experiments were done on a single core of a TPUv3-8 machine using Adam optimizer with a linear warmup for 10 epochs followed by a cosine decay. Random gain, additive gaussian noise, random time masking and mixup (Zhang et al., 2017) augmentations were used for all supervised models. Evaluation was done on the entire audio clip at once, and 95% confidence intervals are reported. No class rebalancing is applied for any dataset.

A.3.1 SER ON AUDIOSET

Training was done on 5-sec random crops, with a 10% random holdout set was used for setting the hyperparameters, after which the model was trained on the entire training set. 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 and 128 for AudioSet balanced setting. mAP, mAUC and *dprime* metrics are used for monitoring performance, and are calculated on the entire audio clip. Results can be found in tables A.2 and A.3.

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 A.2: AudioSet Supervised Learning performance. Pretraining indicates whether contrastive pretrained weights were used

A.3.2 IN-DOMAIN TRANSFER LEARNING: SER ON FSD50K

Training was done on 2.5-sec random crops, and the validation set was used for hyperparameter tuning. A per-tpu-core batch size of 64 worked best. mAP and *dprime* metrics are reported (Tables A.4, A.5).

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., 2019)	-	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 A.3: AudioSet SER performance with comparable baselines. Confidence intervals omitted.

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 A.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 A.5: FSD50k eval performance v/s comparable baselines. Confidence intervals omitted.

A.3.3 Out-of-domain transfer learning: Stopword Identification on the Speech Commands(v2) dataset

All audio samples are ≤ 1 sec in duration, so no random cropping was done while training. A batch size of 128 was used for all experiments. Table A.6 reports classification accuracy of the proposed feature encoder as well as recent comparable baselines.

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 A.6: Speech Commands v2 Test Accuracy. Recent comparable baselines for reference. Parenthesis indicate pretrained weights used in the proposed work (bold)

It is evident from the results on the above mentioned benchmark datasets that while the proposed method doesn't achieve state-of-the-art performance, it is sufficiently performant for the purpose of the study.

B Analyzing layer importances

We explore how layers from the contrastive and the supervised model contribute to recognition performance on the FSD50K and the Speech Commands validation set by incrementally increasing retained layers, followed by temporal average pooling and adding a classifier on top. We conducted two separate experiments, fine-tuning the entire model (*full*) and training just the classifier added on top (*fconly*).

Figure 4 shows steady improvement as we add layers for both contrastive and supervised pretraining in both settings. 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, 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 (right), 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.

```
def sonification(f_enc, deconvnets, x, L, f_map_index):
    """
    Pseudocode for Sonification in Pythonic syntax
    Parameters
    ----------
        f_enc: CNN12 feature encoder
        deconvnets: dictionary with DeconvNet for every layer such that
            key -> block index
            value -> DeconvNet corresponding to the block
        x: input raw waveform signal
        L: {conv-relu-[mp]} block of interest, 1 <= L <= 11
        f_map_index: index of feature map to inspect
    Returns
    -------
        s: sonification signal
    """
    feature_maps, switch_indices = f_enc(x)
    # where feature_maps and switch indices are
    # named key:value pairs corresponding to all layers in the block
    temp = feature_maps[L]
    # where zeros_like creates placeholder tensor of same shape
    current_map = zeros_like(temp)
    # setting all feature maps except f_map_index to zero
    current_map[f_map_index] = temp[f_map_index]
    curr_block_index = L
    while curr_block_index >= 1:
        # get deconvnet for current block
        deconv_curr_block = deconvnets[curr_block_index]
        # apply current deconvnet
        if switch_indices[curr_block_index]:
            # Applying Max Unpooling operation
            current_map = deconvnet.max_unpool(current_map,
                switch_indices[curr_block_index])
        current_map = relu(current_map)
        # Transposed Convolution layer with shared weights
        current_map = deconvnet.transposed_convolution(current_map)
        curr_block_index -= 1
    # we get sonification signal in the input space
    s = current_map
    # Scale s to have unit maximum magnitude
    s = max_abs_scale(s, min=-1, max=1)
    # s is scaled to {-max|x|, max|x|}
    s *= max(abs(x))
    return s
```

Figure B.2: Sonification Pseudo-code

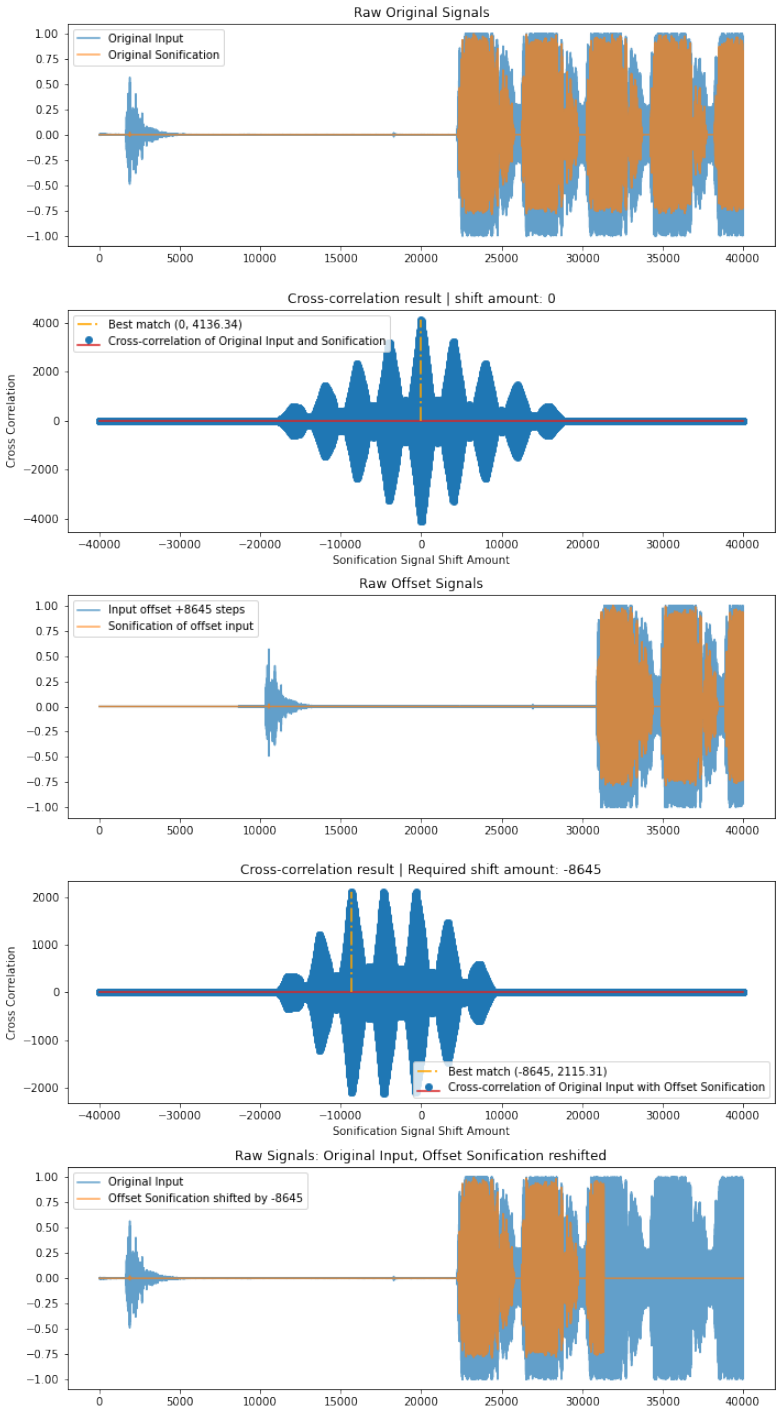

Figure B.3: Time-invariance of the input and corresponding sonifications. In the figure, sonification of an input offset by (randomly selected) +8645 steps is centered with the original input by left-shifting it by the same amount. Other samples demonstrated the same