Composing General Audio Representation by Fusing Multilayer Features of a Pre-trained Model

Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino
NTT Communication Science Laboratories
NTT Corporation
Atsugi, Japan
E-mail: daisuke.niizumi.dt@hco.ntt.co.jp

***Abstract*—Many application studies rely on audio DNN models pre-trained on a large-scale dataset as essential feature extractors, and they extract features from the last layers. In this study, we focus on our finding that the middle layer features of existing supervised pre-trained models are more effective than the late layer features for some tasks. We propose a simple approach to compose features effective for general-purpose applications, consisting of two steps: (1) calculating feature vectors along the time frame from middle/late layer outputs, and (2) fusing them. This approach improves the utility of frequency and channel information in downstream processes, and combines the effectiveness of middle and late layer features for different tasks. As a result, the feature vectors become effective for general purposes. In the experiments using VGGish, PANNs' CNN14, and AST on nine downstream tasks, we first show that each layer output of these models serves different tasks. Then, we demonstrate that the proposed approach significantly improves their performance and brings it to a level comparable to that of the state-of-the-art. In particular, the performance of the non-semantic speech (NOSS) tasks greatly improves, especially on Speech commands V2 with VGGish of $+77.1$ (14.3% to 91.4%).**

***Index Terms*—pre-trained model, feature fusion, global pooling, general-purpose audio representation**

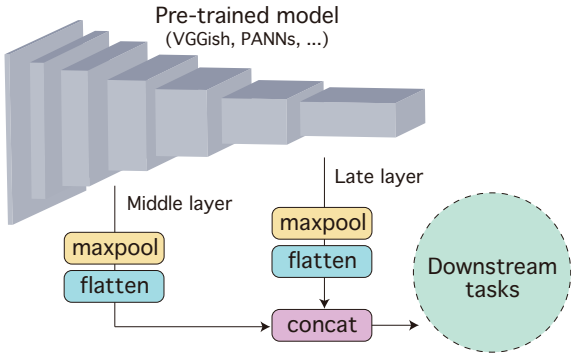

Fig. 1: Proposed feature calculation flow. The number of time frames of layer outputs is adjusted by *maxpool*, and the channel and frequency axes are flattened to get feature vectors along time. Then, both middle and late features are concatenated to get the final feature embeddings, making the features effective in general audio downstream tasks. The middle and late layers are chosen based on the layer-wise performance for the tasks.

I. Introduction

Pre-trained models are essential building blocks as feature extractors to transfer learned representations from large-scale datasets. In the audio domain, we can find many applications using pre-trained models: VGGish [1] pre-trained on YouTube-8M [2] are used in conservation monitoring [3], audio captioning [4], and speech emotion recognition [5]; and PANNs [6] pre-trained on AudioSet [7] are used in heart sound classification [8] and conservation monitoring [9].

These applications use models as feature extractors without additional training, but supervised learning models are known to specialize in the pre-training dataset domain [10]. On the other hand, several self-supervised learning models proposed recently show well-balanced performance on general tasks [11] [12] [13]. These models learn general-purpose audio representations and are considered more suitable as feature extractors.

In this study, we focused on the potential performance of middle and late layer features of supervised pre-trained models for various tasks. In our preliminary experiments, the late layer features of supervised pre-trained models, used in the typical applications, performed better than self-supervised pre-trained models on sound event recognition (SER) tasks (e.g., ESC-50 [14], UrbanSound8K [15]), while they performed poorly on other tasks. Surprisingly, however, features from the middle layer performed differently, worse on SER tasks and better on other tasks. Our research question is: *Can we put the strength of both features together for general purpose?*

We think that the reason for the imbalanced performance of the supervised learning model could be the training metric and network architecture. Since the pre-training metric is large-dataset classification accuracy, the late layers are considered specialized to the pre-trained dataset domain [10]. In addition, many of the models based on the image domain network [1] [16] are not designed to process time-frequency (TF) audio input effectively for audio downstream use.

To address the aforementioned problems, we propose a simple approach to calculate general-purpose features by using the outputs from the middle and late layers of a supervised pre-trained model. In addition to the proposed approach for general-purpose feature calculation, our contributions include showing that the effective layers of the supervised pre-trained

model are task-dependent and validating in experiments that the performance of these models can be significantly improved by using the proposed approach.

Our code is available at https://github.com/nttcslab/composing-general-audio-repr.

II. Related Work

Feature computation of pre-trained models Existing audio pre-trained models compute feature vectors similarly to how image domain models do. VGGish [1] and OpenL3 [17] flatten all axes (frequency, time, and channel) of the convolutional layer output into a single embedding vector. COLA [11] and TRILL [18] apply global averaging or max pooling and summarize frequency and time frame axes.

These operations can be problems for downstream tasks. For example, the pitch of the voice is considered necessary for the speaker recognition task; thus, frequency-wise information is essential. While voice inflection is vital for speech emotion recognition; thus time-wise information is needed.

While flattening preserves all the information, it is difficult to calculate feature statistics (e.g., averaging frequency bins temporally) from flattened vectors. On the other hand, global pooling makes information per frequency or time frame unavailable in the later processes. Both of these issues could impair the utility of the feature vectors in downstream tasks.

General-purpose audio representations Self-supervised learning models such as COLA [11], BYOL-A [12], and Slowfast NFNets [13] have been proposed for general-purpose or universal audio representations pre-trained on AudioSet without labels. In experiments, these models generally demonstrate well-balanced performance in tasks.

Multilevel feature fusion In multimodal application research, feature fusion of multilevel (layer) outputs is utilized. For example, [19] fuses multilayer features from video and audio encoders. In the image domain, [20] uses the size transformation function to match the feature size of each layer to combine multilayer features. In the audio domain, AudioCaps [21] evaluates various audio features, including combinations of multi-layer outputs for the audio captioning task, but not for other tasks.

III. Proposed Approach

Our approach to improve performance of models for general tasks consists of calculating feature vectors along the time frame from layer outputs and fusing middle and late layer feature vectors. Fig. 1 illustrates the calculation flow.

To calculate feature vectors along the time frame, we first adjust the number of time frames to a T_o using *maxpool* and then flatten the channel and frequency along time. Adjusting the time frame of any layer feature to a T_o enables subsequent *fusion*, whereas flattening transforms the channel and frequency into vectors without the information loss that could be caused by averaging or max operations found in conventional methods:

$$\hat{z_l} = \mathit{flatten}(\mathit{maxpool}(z_l, T_o)), \tag{1}$$

where $z_l \in R^{B \times C_l \times F_l \times T_l}$ is the lth layer output, and B, C_l, F_l, and T_l are the batch size, number of channels, number of frequency bins, and number of time frames, respectively. The kernel and stride parameters of *maxpool* are set to reduce T_l to T_o. As a result, $\hat{z_l} \in R^{B \times C_l F_l \times T_o}$ is calculated as a feature vector with the time frame adjusted to T_o.

In the conventional calculation with TF features as input, flattening all axes transforms features to $R^{B \times C_l F_l T_l}$, making it difficult to use the features in later processes such as calculating the statistics of frequency along time. Another problem is that the global averaging or max pooling transforms features to $R^{B \times C_l}$. As a result, frequency and time information is no longer available for downstream tasks. The calculation by Eq. (1) solves these problems by preserving the information for all axes and simplifying the usage of feature vectors for each time frame.

The feature vectors are fused as follows:

$$z = \mathit{concat}(\hat{z_M}, \hat{z_L}), \tag{2}$$

where $\hat{z_M}$ and $\hat{z_L}$ are the features from middle layer M and late layer L calculated by the Eq. (1), and $z \in R^{B \times (C_M F_M + C_L F_L) \times T_o}$ is the fused feature vector along time. This calculation concatenates feature vectors from the middle and late layers for each time frame, which preserves all the available information from different layers.

Layer M and L are chosen based on the layer-wise performance for the tasks. We observed in preliminary experiments that the late layers of supervised pre-trained models excel on a set of downstream tasks D_L, whereas the middle layers perform better on other set of tasks D_M. We choose the middle layer M, which shows the best average performance for D_M, and the late layer L, which shows the best average performance for D_L.

While z provides the feature vector per time frame, the following from PANNs [6] calculates temporal statistics to make a single vector for variable-length audio.

$$\tilde{z} = \mathit{mean}(z) + \mathit{max}(z) \tag{3}$$

This summarizes the time axis as combined statistics of mean and max operation, and it has performed well in previous studies [6] [12]. We used the embedding vector $\tilde{z} \in R^{B \times (C_M F_M + C_L F_L)}$ in the following experiments.

IV. Experiments

Here, we show that the performance of each layer of existing supervised learning models are task-dependent in Section IV-B. Next, we evaluate performance improvement of these models using our approach in Section IV-C. Then, we compare our approach with SOTA in Section IV-D.

We conducted a linear evaluation using three models and nine downstream tasks. The linear evaluation tests the effectiveness of the features of the pre-trained models by training a linear model that takes as input the features, and the test accuracy is the result.

A. Experimental Details

Linear evaluation details To train the linear model, we used the validation set for early stopping with a patience of 20 epochs and trained for up to 200 epochs with the Adam optimizer. We manually tuned the learning rate to get the best results between 0.00001 and 0.01 for every test. We ran each evaluation three times and averaged the results.

TABLE I: VGGish layers. The ReLU layer output shape [(B)atch, (C)hannel, (T)ime, (F)requency] is calculated to [(B)atch, (D)imension, (T)ime] by Eq. (1).

Layer #	Operation	Parameters/Output shape
1	Conv	(1, 64, kernel=(3, 3), stride=(1, 1))
2	ReLU	$[B, 64, 96, 64] \rightarrow$ Eq. (1) $\rightarrow [B, 4096, 6]$
3	MaxPool	(kernel=2, stride=2)
4	Conv	(64, 128, kernel=(3, 3), stride=(1, 1))
5	ReLU	$[B, 128, 48, 32] \rightarrow$ Eq. (1) $\rightarrow [B, 4096, 6]$
6	MaxPool	(kernel=2, stride=2)
7	Conv	(128, 256, kernel=(3, 3), stride=(1, 1))
8	ReLU	$[B, 256, 24, 16] \rightarrow$ Eq. (1) $\rightarrow [B, 4096, 6]$
9	Conv	(256, 256, kernel=(3, 3), stride=(1, 1))
10	ReLU	$[B, 256, 24, 16] \rightarrow$ Eq. (1) $\rightarrow [B, 4096, 6]$
11	MaxPool	(kernel=2, stride=2)
12	Conv	(256, 512, kernel=(3, 3), stride=(1, 1))
13	ReLU	$[B, 512, 12, 8] \rightarrow$ Eq. (1) $\rightarrow [B, 4096, 6]$
14	Conv	(512, 512, kernel=(3, 3), stride=(1, 1))
15	ReLU	$[B, 512, 12, 8] \rightarrow$ Eq. (1) $\rightarrow [B, 4096, 6]$
16	MaxPool	(kernel=2, stride=2)
	(flatten)	$[B, 12288]$
17	Linear	(in=12288, out=4096)
18	ReLU	$[B, 4096] \rightarrow$ repeat $\rightarrow [B, 4096, 6]$
19	Linear	(in=4096, out=4096)
20	ReLU	$[B, 4096] \rightarrow$ repeat $\rightarrow [B, 4096, 6]$
21	Linear	(in=4096, out=128)
22	ReLU	$[B, 128] \rightarrow$ repeat $\rightarrow [B, 128, 6]$

TABLE II: CNN14 convolutional blocks. Block output shape [B, C, T, F] is calculated to [B, D, T] by Eq. (1).

Block #	Output shape
1	$[B, 64, T/2, 32] \rightarrow$ Eq. (1) $\rightarrow [B, 2048, T/32]$
2	$[B, 128, T/4, 16] \rightarrow$ Eq. (1) $\rightarrow [B, 2048, T/32]$
3	$[B, 256, T/8, 8] \rightarrow$ Eq. (1) $\rightarrow [B, 2048, T/32]$
4	$[B, 512, T/16, 4] \rightarrow$ Eq. (1) $\rightarrow [B, 2048, T/32]$
5	$[B, 1024, T/32, 2] \rightarrow$ Eq. (1) $\rightarrow [B, 2048, T/32]$
6	$[B, 2048, T/32, 2] \rightarrow$ Eq. (1) $\rightarrow [B, 2048, T/32]$

Downstream tasks We employed nine tasks widely used in previous studies [6] [11] [17] [18] [16]: ESC-50 [14] (environmental sound classification) and UrbanSound8K (US8K, urban sound classification) [15] from sound event recognition (SER) tasks; Speech Commands V2 [22] (SPCV2, speech command word classification), VoxCeleb1 [23] (VC1, speaker identification), VoxForge [24] (VF, language identification), and CREMA-D [25] (CRM-D, speech emotion recognition) from non-semantic speech (NOSS) tasks; and GTZAN [26] (music genre recognition), NSynth [27] (music instrument family classification) and Pitch Audio Dataset (Surge synthesizer) [28] (Surge, pitch audio classification) from music tasks.

Pre-trained models We tested three models: VGGish [1] and CNN14 from PANNs [6], which are CNN architectures, and Audio Spectrogram Transformer (AST) [16], which is a Transformer architecture. The followings describe their details.

1) VGGish: Table I shows VGGish layers, which consists a stack of convolutional layers followed by three fully connected (FC) layers, 22 layers in total. This model flattens all axes before the 17th layer. Since the input time frame length is fixed to $T = 96$, we converted the variable length inputs into feature vectors as follows: encode all the divided segments of length T of a input into feature vectors, concatenate feature vectors along time, then apply Eq. (3) to get a single vector for the input. We use $T_o = 6$ which is the number of time frames of 16th layer output. The layer 18, 20, and 22 outputs don't have time axis, then we repeat them T_o times to form the time axis. We evaluate all ReLU layer outputs at layers $\in \{2, 5, 8, 10, 13, 15, 18, 20, 22\}$.

2) PANNs' CNN14: Table II shows convolutional blocks of CNN14. CNN14 accepts input with variable length T, and we set $T_o = T/32$. We evaluated all the block outputs.

3) AST: AST is a Transformer model with 12 layers, and we evaluated layer 2 to 12 outputs using 768-d `[CLS]` token embeddings. These embeddings are vectors without a time axis, unlike in CNN models.

B. Evaluating Layer-wise Performance

In this experiment, we evaluated the performance of each layer for all models. The output of each layer was transformed into feature vectors per time frame using Eq. (1), and into a single vector using Eq. (3). Fig. 2, 3, 4 show the results for VGGish, CNN14, and AST, respectively.

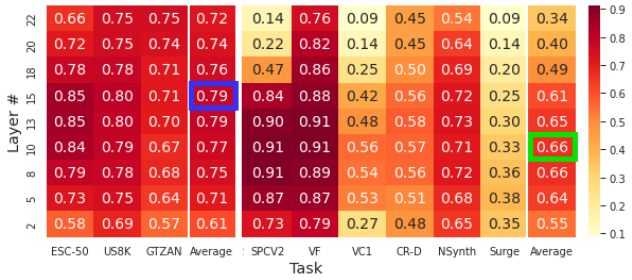

Fig. 2: VGGish layer-wise evaluation accuracies (%).

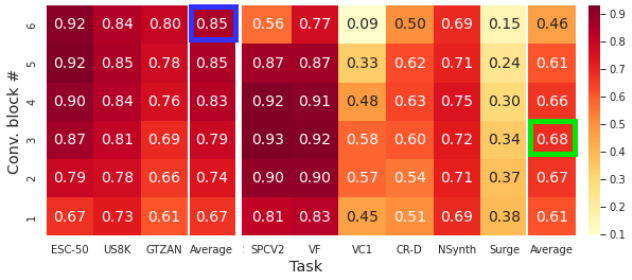

Fig. 3: PANNs' CNN14 layer-wise evaluation accuracies (%).

The results show that the peaks of performance in a layer are different for each task. For the ESC-50/US8K/GTZAN, the peaks are in the late layers, while for the other tasks, they are in the middle layers. This is also clearly shown by

TABLE III: Pre-trained model accuracy improvements (%) achieved by the proposed approach.

	SER tasks		NOSS tasks				Music tasks			
Representation	ESC-50	US8K	SPCV2	VC1	VF	CRM-D	GTZAN	NSynth	Surge	Avg.
VGGish [1]	68.2	75.1	14.3	9.0	75.7	44.4	75.3	53.9	8.8	47.2
VGGish-Fusion#10#15	86.5	80.9	91.4	54.5	91.6	59.3	70.8	73.6	33.3	71.3
difference	+18.2	+5.8	+77.1	+45.5	+16.0	+15.0	-4.5	+19.7	+24.5	+24.1
CNN14 [1]	90.1	82.0	51.4	8.0	75.0	50.7	79.7	66.0	10.4	57.0
CNN14-Fusion#3#6	93.0	85.8	91.3	50.6	90.5	59.0	77.4	73.8	32.4	72.6
difference	+2.9	+3.8	+39.9	+42.7	+15.5	+8.3	-2.3	+7.8	+22.0	+15.6
AST [1]	93.5	85.5	71.8	16.5	81.2	57.9	84.3	73.2	25.8	65.5
AST-Fusion#5#12	94.2	85.5	80.4	24.9	87.6	60.7	82.9	77.6	34.6	69.8
difference	+0.6	+0.0	+8.6	+8.4	+6.4	+2.8	-1.4	+4.5	+8.9	+4.3

[1] The baseline results used the last layer features from the original models.

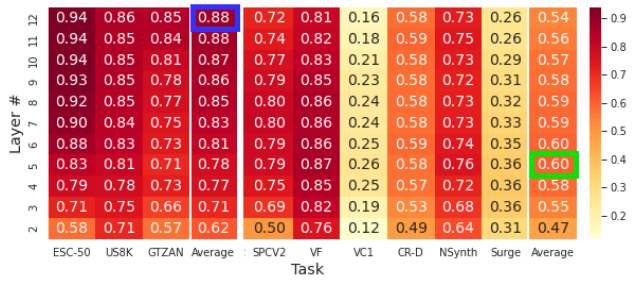

Fig. 4: AST layer-wise evaluation accuracies (%).

comparing the average peaks for the ESC-50/US8K/GTZAN tasks in the blue box with the average peak for other tasks in the green box.

Focusing on the layer-wise results, we see that late layers perform well on ESC-50/US8K/GTZAN, while on the others, especially NOSS tasks such as VC1, they perform quite poorly, showing a substantial gap between tasks. On the other hand, middle layers with the green box perform worse on ESC-50/US8K/GTZAN. While no single layer satisfies all task performances, the late layers perform more imbalanced.

The VGGish results show another problem: the performance drops after layer #15. One possible reason is that late layers are specialized to the pre-training dataset. Another reason could be the difference in calculation; while features up to #15 are calculated using Eq. (1), the features after #15 are calculated by flattening all axes. Many VGGish application studies use the feature from late layer #22 (FC2) [3] [4] [5]; however, other layer features calculated by using Eq. (1) calculation potentially become more effective for these applications.

C. Evaluating Proposed Approach

In this experiment, we applied the proposed approach to VGGish, CNN14, and AST to evaluate the performance improvement.

The middle layer M and late layer L were determined for each model using the results of Section IV-B. For L, we choose the layer in the blue box, where the peaks of the average result for ESC-50/US8K/GTZAN, and for M, we choose the layer in the green box, where the peaks of the average result for other tasks. Thus, we used $M = 10$ and $L = 15$ for VGGish, denoting the audio representation calculated by the proposed approach as VGGish-Fusion#10#15. The same goes for CNN14-Fusion#3#6 for PANNs' CNN14 with $M = 3$ and $L = 6$, and AST-Fusion#5#12 for AST with $M = 5$ and $L = 12$.

Table III compares the results before and after the application of the proposed approach and shows a significant improvement in performance for all models. In particular, the performance was greatly improved in the NOSS task, especially SPCV2, which showed a significant improvement of +77.1 from 14.3% to 91.4% with VGGish, and improvement of +39.9 from 51.4% to 91.3% in CNN14.

In addition, the CNN models (VGGish, CNN14) also improved the performance of Surge (pitch classification) significantly, suggesting the contribution of local features. The earlier the CNN layer is, the higher the frequency resolution becomes, which could make it easier to detect the pitch.

The similar performance improvements of AST in NOSS and Surge tasks show that the proposed approach is also effective for the Transformer architecture. As a previous study [29] showed that early layers attend both locally and globally, the local feature in the fused earlier layer output possibly contributed to the improvements, similar to how the CNN layer does.

The performance of the GTZAN task slightly degraded for all models, indicating that the proposed approach can also cause degradation. However, this degradation is small compared to the overall performance improvements; thus, we think the proposed approach is generally beneficial.

D. Comparison with State of the Art

The results shown in Table IV indicate that the proposed approach brings the performance of the existing models to a level comparable with that of SOTA. It improves the inferior NOSS task performance while maintaining the SER task performance at a higher level than that of SOTA models.

These results suggest that existing supervised pre-trained models have sufficient performance potential, which the proposed approach can exploit. We think that the improved audio representations could generally serve various tasks.

TABLE IV: Comparison with state of the art models (%).

	SER tasks		NOSS tasks				Music tasks		
Representation	ESC-50	US8K	SPCV2	VC1	VF	CRM-D	GTZAN	NSynth	Surge
SF-NFNet-F0 [13]	91.1	N/A	**93.0**	**64.9**	90.4	N/A	N/A	**78.2**	N/A
COLA [11]	N/A	N/A	62.4	29.9	71.3	N/A	N/A	63.4	N/A
OpenL3 [17] [1]	79.8	79.3	87.9	40.7	90.1	60.4	73.3	75.6	**36.4**
BYOL-A [12] [1]	83.7	79.1	92.2	40.1	90.2	**62.8**	73.6	74.1	26.2
VGGish-Fusion#10#15	86.5	80.9	91.4	54.5	**91.6**	59.3	70.8	73.6	33.3
CNN14-Fusion#3#6	93.0	**85.8**	91.3	50.6	90.5	59.0	77.4	73.8	32.4
AST-Fusion#5#12	**94.2**	85.5	80.4	24.9	87.6	60.7	**82.9**	77.6	34.6

[1] Underlined results were obtained in this study using publicly available pre-trained models.

V. Conclusion

In this paper, we proposed an approach to improve feature calculation of existing supervised pre-trained models without fine-tuning, and showed that the resulting features could serve as general-purpose audio representations.

The proposed approach first calculates feature vectors aligned with the time frame to improve the utility of frequency and channel information in downstream processes. Then, it fuses feature vectors from the middle and late layers, combining the effectiveness of these features for different tasks.

In the experiments using VGGish, PANNs ' CNN14, and AST on nine downstream tasks, we showed that each layer output from these models serves different tasks, and showed that the proposed approach significantly improves performance and brings it to a level comparable to that of SOTA models. Particularly, the performance of the NOSS tasks greatly improves, especially on SPCV2 with VGGish of $+77.1$ (14.3% to 91.4%), while maintaining higher performance on SER tasks.

Our proposed approach provides a simple way to exploit existing supervised pre-trained models as general-purpose audio representations. It could make future audio application studies achieve better performance. Our code is available online.

References

[1] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "Cnn architectures for largescale audio classification," in *ICASSP*, 2017, pp. 131–135.

[2] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[3] S. S. Sethi, N. S. Jones, B. D. Fulcher, L. Picinali, D. J. Clink, H. Klinck, C. D. L. Orme, P. H. Wrege, and R. M. Ewers, "Characterizing soundscapes across diverse ecosystems using a universal acoustic feature set," *Proceedings of the National Academy of Sciences*, 2020.

[4] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A transformer-based audio captioning model with keyword estimation," in *Interspeech*, Oct 2020.

[5] W. Jiang, Z. Wang, J. S. Jin, X. Han, and C. Li, "Speech emotion recognition with heterogeneous feature unification of deep neural network," *Sensors*, vol. 19, no. 12, 2019.

[6] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 28, pp. 2880–2894, 2020.

[7] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *ICASSP*, 2017, pp. 776–780.

[8] T. Koike, K. Qian, Q. Kong, M. D. Plumbley, B. W. Schuller, and Y. Yamamoto, "Audio for audio is better? an investigation on transfer learning models for heart sound classification," in *EMBC*, 2020.

[9] I. Tolkova, B. Chu, M. Hedman, S. Kahl, and H. Klinck, "Parsing birdsong with deep audio embeddings," *arXiv:2108.09203*, 2021.

[10] H. Maennel, I. M. Alabdulmohsin, I. O. Tolstikhin, R. Baldock, O. Bousquet, S. Gelly, and D. Keysers, "What do neural networks learn when trained with random labels?" in *NeurIPS*, 2020.

[11] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *ICASSP*, Jun 2021.

[12] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Self-supervised learning for general-purpose audio representation," in *IJCNN*, Jul 2021.

[13] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J.-B. Alayrac, S. Dieleman, J. Carreira, and A. van den Oord, "Towards learning universal audio representations," *arXiv preprint arXiv:2111.12124*, 2021.

[14] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *ACM-MM*, 2015, pp. 1015–1018.

[15] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *ACM-MM*, Nov. 2014, pp. 1041–1044.

[16] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," *Interspeech 2021*, Aug 2021.

[17] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen and learn more: Design choices for deep audio embeddings," in *ICASSP*, Brighton, UK, May 2019, pp. 3852—3 856.

[18] J. Shor, A. Jansen, R. Maor, O. Lang, O. Tuval, F. d. C. Quitry, M. Tagliasacchi, I. Shavitt, D. Emanuel, and Y. Haviv, "Towards learning a universal non-semantic representation of speech," in *Interspeech*, 2020.

[19] X. Xu, Y. Wang, D. Xu, Y. Peng, C. Zhang, J. Jia, Y. Wang, and B. Chen, "Mffcn: Multi-layer feature fusion convolution network for audio-visual speech enhancement," *arXiv preprint arXiv:2101.05975*, 2021.

[20] C. Ma, X. Mu, and D. Sha, "Multi-layers feature fusion of convolutional neural network for scene classification of remote sensing," *IEEE Access*, vol. 7, pp. 121 685–121 694, 2019.

[21] C. D. Kim, B. Kim, H. Lee, and G. Kim, "Audiocaps: Generating captions for audios in the wild," in *NAACL-HLT*, 2019.

[22] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *arXiv preprint arXiv::1804.03209*, Apr. 2018.

[23] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Interspeech*, 2017, pp. 2616–2620.

[24] K. MacLean, "Voxforge," 2018. [Online]. Available: http://www.voxforge.org/home

[25] H. Cao, D. G. Cooper, M. K. Keutmann, R. C. Gur, A. Nenkova, and R. Verma, "Crema-d: Crowd-sourced emotional multimodal actors dataset," *IEEE Trans. Affective Comput.*, pp. 377–390, 2014.

[26] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Speech Audio Process.*, vol. 10, no. 5, 2002.

[27] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *ICML*, 2017, pp. 1068–1077.

[28] J. Turian, J. Shier, G. Tzanetakis, K. McNally, and M. Henry, "One billion audio sounds from GPU-enabled modular synthesis," in *DAFx2020*, Sep. 2021.

[29] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do vision transformers see like convolutional neural networks?" in *NeurIPS*, 2021.