Interspeech 2022
18-22 September 2022, Incheon, Korea

MAE-AST: Masked Autoencoding Audio Spectrogram Transformer

Alan Baade, Puyuan Peng, David Harwath

Department of Computer Science, The University of Texas at Austin
abaade,pyp,harwath@utexas.edu

Abstract

In this paper, we propose a simple yet powerful improvement over the recent Self-Supervised Audio Spectrogram Transformer (SSAST) model for speech and audio classification. Specifically, we leverage the insight that the SSAST uses a very high masking ratio (75%) during pretraining, meaning that the vast majority of self-attention compute is performed on mask tokens. We address this by integrating the encoder-decoder architecture from Masked Autoencoders are Scalable Vision Learners (MAE) into the SSAST, where a deep encoder operates on only unmasked input, and a shallow decoder operates on encoder outputs and mask tokens. We find that MAE-like pretraining can provide a 3× speedup and 2× memory usage reduction over the vanilla SSAST using current audio pretraining strategies with ordinary model and input sizes. When fine-tuning on downstream tasks, which only uses the encoder, we find that our approach outperforms the SSAST on a variety of downstream tasks. We further conduct comprehensive evaluations into different strategies of pretraining and explore differences in MAE-style pretraining between the visual and audio domains. Code at https://github.com/AlanBaade/MAE-AST-Public

Index Terms: audio classification, self-attention, Transformer, self-supervised

1. Introduction and Related Work

Training high-quality neural networks requires lots of data. However, simple-to-train-with labeled data is expensive and limited, while unlabeled data from the internet is far more abundant. To address this, self-supervised learning techniques have seen an explosion in research interest, especially after producing incredible results in the natural language domain [1, 2, 3].

Recently, the success of self-supervised pretraining has migrated into the audio [4, 5] and vision [6, 7] communities. Of particular interest to this paper is the Self-Supervised Audio Spectrogram Transformer (SSAST) [8], a self-supervised extension to the Audio Spectrogram Transformer (AST) [9]. The SSAST introduced the first patch-based and fully self-attention [10] based pretraining strategies for audio event classification, and achieved remarkable results for both audio event classification and several speech classification tasks. However, one common criticism of Transformer models is their massive computational overhead, in large part due to their $O(N^2)$ memory and time complexity when computing the self-attention operation. Many so-called "efficient" Transformers have been proposed [11] that try to mitigate this cost. Of particular relevance to us is the Masked Autoencoder [12], which completely discards masked input tokens during the encoding step, resulting in significant increases in computational efficiency.

In this paper, we improve upon the SSAST architecture by incorporating ideas from the Masked Autoencoder (MAE) introduced by Kaiming et al. [12], which asymmetrically applies BERT-like [1] pretraining to the visual domain with an encoder-decoder architecture. Specifically, we observe that we can reduce the input length to a majority of transformer layers by taking advantage of the high masking ratio used for audio pretraining (75%). We additionally notice that, similar to the visual domain, reconstruction in the audio spectrogram domain introduces generally unwanted, low-level pixel information to the final layers of pretrained models. From these observations, we propose the Masked Autoencoding Audio Spectrogram Transformer (MAE-AST), which runs only unmasked tokens through a large encoder and then concatenates mask tokens with encoder output embeddings before feeding them to a shallow decoder. Fine-tuning on downstream tasks is then performed using only the encoder, removing the decoder's reconstruction layers.

We find two key results: 1) The MAE-AST pretrains using significantly less time (3×) and memory (2×) than the SSAST for a similar model architecture. 2) The MAE-AST outperforms the SSAST under a shared encoder depth on several downstream tasks, all other factors held constant. Additionally, we find that the MAE-AST performs well with only a generative objective, while the SSAST sees a significant performance drop. We further run several experiments on methods to pretrain the MAE-AST and compare MAE-like pretraining on audio to both SSAST-like pretraining on audio and MAE-like pretraining on vision.

2. Masked Autoencoding Audio Spectrogram Transformer

Figure 1 shows the MAE-AST architecture. In this section, we first outline the forward pass, followed by training practices, masking strategies, and the loss function.

2.1. Model Architecture

Input Processing. Identical to the SSAST, we take as input a 16khz audio waveform and convert it into 128-dimensional log Mel filterbank features with a frame length of 25 ms and frame shift of 10 ms. Like AST and SSAST, we normalize the input to mean 0 and standard deviation ½ . This spectrogram is then split when patch-based into a series of 16 filter × 16 frame tokens and when frame-based into 128 filter × 2 frame tokens. We then apply a masking method, which splits the input patches or frames into masked and unmasked subsets. We experiment with several masking strategies, discussed later.

Positional Embeddings. Consistent with [12], we use fixed sinusoidal positional embeddings for both patch-based and frame-based tokenization. As the input spectrogram data is only variable-length in time, we use one-dimensional positional embeddings for both tokenizations, with patch input being unfolded into a one-dimensional stream ordered first by channel and second by time to support variable input lengths, as shown in Figure 1.

Encoder. The encoder follows a standard ViT [6] architec-

Copyright © 2022 ISCA

10.21437/Interspeech.2022-10961

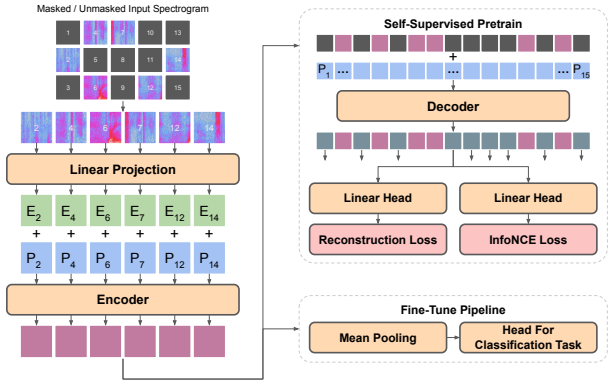

Figure 1: *The architecture of the MAE-AST for patch-based inputs. First, the input spectrogram of size* $128 \times 100t$ *is split into* 16×16 *pixel patches. For pretraining, a mask is then applied to some tokens. Unmasked patches are unrolled first by channel dimension and then by time dimension and are embedded into a vector via linear projection. These unmasked patch embeddings are added to 1D sinusoidal positional embeddings and fed into the encoder. During pretraining, mask tokens are combined with encoder outputs and positional embeddings are added to all tokens before being sent through the decoder. The joint discriminative and generative loss is calculated on the decoder outputs for mask tokens. For fine-tuning, the encoder outputs are mean pooled to create a representation vector and the decoder isn't touched.*

ture. Unlike BERT-like models, the MAE encoder only operates on unmasked tokens. We flatten the unmasked input patches or frames via a linear projection into a 768-dimensional latent vector. We then add positional embeddings to these latents and input the summed vectors to the encoder.

Decoder. The decoder is only used during pretraining and is composed of standard transformer encoder blocks. Both the encoder output and mask tokens are sent to the decoder, with mask tokens at the decoder's input being represented by a shared, learned embedding. We add sinusoidal positional embeddings to all tokens before inputting into the decoder. The output embeddings of masked tokens from the decoder are then sent into two single-layer linear projections, one for each loss function, that map to a vector the size of the patch input. Specifics of calculating loss are discussed later.

By default, we use an encoder with 6 layers, and a decoder of 2 layers. Both the encoder and decoder use 12 heads and a width of 768. Following the MAE [12], we throw away the decoder during fine-tuning, training solely with the encoder. We mean pool the encoder output last hidden states to create a single vector representation for classification tasks.

To create a fair comparison between MAE-like and traditional BERT-like architectures for audio pretraining, we largely adopt the input pipelining and loss calculation techniques from the SSAST. However, we make two minor changes:

1) During training, the original AST paper [9] applies overlap to its input patches for a modest performance improvement. The SSAST paper does not apply overlap during pretraining to prevent the model from using it to cheat the reconstruction task. However, SSAST adds overlap back during fine-tuning to match the original AST paper via interpolating learned positional embeddings. To match the MAE paper, we forgo overlap during both pretraining and fine-tuning.

2) The SSAST uses interpolation or truncation of positional embeddings to support variable-length inputs during fine-tuning. To match the MAE paper, we make no modifications to our positional embeddings for fine-tuning.

We perform experiments under the fairseq library [13] using the Adam [14] optimizer with weight decay 0.01, initial learning rate 0.0001, and a polynomial decay learning rate scheduler. With fairseq, we use a maximum number of tokens per GPU that is roughly equivalent to a batch size of 32. Training runs for up to 600,000 iterations, which provides the same number of audio clips during pretraining as SSAST. We train using one NVIDIA Quadro RTX-8000, and pretraining takes about two days.

2.2. Mask Sampling

Mask sampling for pretraining generally involves two considerations, the percent of tokens to mask and the amount of chunking between masked tokens. Masking strategies with too little chunking or too few masked tokens will create too easy of a task, allowing the model to simply interpolate missing pixels using their neighbors. The proportion to mask is especially important for MAE-based models, as a higher masking ratio decreases the length of the input to the decoder, which can provide significant speedups due to the quadratic time and memory complexity of Transformers.

We experiment with eight overall masking and input strategies via combinations of the following factors: 1) Patch-Based versus Frame-Based inputs, 2) Fully random versus chunked masking, 3) Different masking ratios. We implement these strategies and relate them to previous work as follows:

1) Patch-Based, Fully Random masking is the strategy implemented by the original MAE paper for image pretraining. Like the MAE paper, we mask by shuffling the input patches and keeping the first $1 - p$ proportion of tokens. 2) Patch-Based, Chunked masking is adapted using the strategy from SSAST, which selects a level of chunking $C \in \{3, 4, 5\}$ and randomly masks $C \times C$ chunks of tokens until a proportion $\geq p$ is masked, then truncating down to p. This unbiased sampling is slow, so we apply the same mask across a batch to reduce compute. 3) Frame-Based, Fully Random masking is the strategy implemented by SSAST for frame-based experiments, and mask selection is performed in the same way as patch-based, fully random masking. 4) Frame-Based, Chunked masking is implemented using the algorithm from Wav2Vec2.0 [4], where spans of length $M = 10$ are masked with each index having a P probability of being chosen to start a span. This method often creates overlap between chunks, making the output length unpredictable. To make fair comparisons between different masking strategies, we select P so that the average proportion of tokens masked is p after accounting for overlap.

2.3. Joint Discriminative and Generative Pretraining

One of the key contributions of the SSAST paper was combining both discriminative (classification) and generative (reconstruction) losses for audio pretraining. Our model continues to adopt this strategy, however with a few small differences.

In the SSAST, the encoder output is fed into two separate two-layer feed-forward neural networks for self-supervised pretraining, one each for reconstruction and classification loss. Because the decoder head of MAE-based models already provides compute past the last layer of embeddings used, we use a single linear layer head for our final output projection for each loss.

We implement loss functions identically to SSAST. Reconstruction loss is calculated as a mean-squared error between the unnormalized output of the linear reconstruction head and the normalized input. Classification loss intends to create a vector c_i similar to the masked input patch or frame x_i but dissimilar to other masked input. Like the SSAST, we obtain negative samples from other masked inputs within the same audio segment, which both helps lessen useless features being encoded (such as microphone type) and reduces the impact of batch size on pretraining. We use the InfoNCE [15] loss for classification. The classification and reconstruction losses are summed and balanced by a factor λ which is multiplied by the reconstruction loss. We use the same $\lambda = 10$ as SSAST.

3. Experiments

3.1. Datasets

Pretraining Datasets. To fairly show that the MAE encoder-decoder architecture is the cause of performance gains, we use the same pretraining datasets as SSAST. The SSAST pretraining data consists of uniformly shuffled data from AudioSet-2M [16] and Librispeech [17] with labels discarded. AudioSet-2M contains two million ten-second audio clips from YouTube, and provides data for both audio events and speech. AudioSet is subdivided into unbalanced, balanced, and eval partitions, with the balanced and eval sections having roughly the same number of examples per classification label. Librispeech provides 960 hours of audiobook data and is included because AudioSet clips do not necessarily contain speech. We pretrain using both the unbalanced and balanced AudioSet partitions in addition to the librispeech train data.

Fine-Tuning Datasets. We fine-tune on the same six commonly-used datasets as SSAST, which provide a variety of downstream audio and speech challenges. We evaluate audio event classification performance by fine-tuning on the AudioSet (AS) and ESC-50 [18] (ESC) datasets. AudioSet provides a general audio classification task, while ESC-50 tests classification of environmental audio. Like the SSAST paper, we fine-tune on only the balanced partition of AudioSet and test on the eval partition. We evaluate speech performance by fine-tuning on Speech Commands 1 and 2 [19] for the task of keyword spotting (KS1, KS2), VoxCeleb [20] for speaker identification (SID), and IEMOCAP [21] for emotion recognition (ER). For AS, ESC, and KS2, we use the fine-tuning pipeline from SSAST. For KS1, SID, and ER we use the SUPERB [22] framework, the same fine-tune setting as SSAST. We use default settings for all models, which matched the results of a learning rate grid search.

AudioSet Preprocessing Because the AudioSet data is downloaded from YouTube directly, videos get deleted and the available dataset decreases in size over time: Our downloaded copy of AudioSet contains 86.4% of the original dataset, while SSAST had 94%. To make our results roughly comparable to other papers, we fill in the missing data per label for the balanced and eval sets by sampling from the unbalanced set using the same sampling method as the AudioSet paper. Assuming no correlation between deleted videos and label type, this acts as a random resample of the balanced and eval sets from all downloaded data given that the available balanced and eval sets are chosen, which is the best we can do with limited data. After resampling, our unbalanced set contains 1,762,073 examples, balanced 22,680, and eval 20,920.

We load pretrained weights of SSAST distributed by [8] and fine-tune and test it on our version of AudioSet, meaning all experiments are evaluated on the same data. Importantly, we do not replicate pretraining. However, this should only serve as a disadvantage to our models as recently downloaded data is a subset of data from earlier downloads, barring a small amount of reuploaded videos.

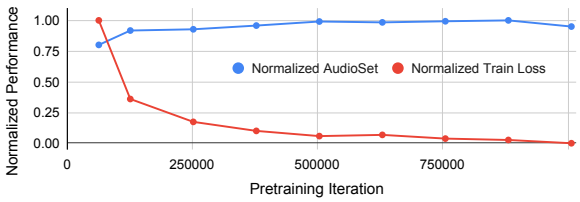

Figure 2: *Convergence of MAE-AST. We train the default model for an extended amount and save a checkpoint at training epochs 1,2,4,6,8,10,12,14, and 16. We evaluate these saves on the downstream AudioSet task, and normalize the results in [0,1]. We find that MAE-AST pretraining converges in 8 epochs (500,000 iterations), the exact same as SSAST. MAE-AST is also non-brittle to increased pretraining time, maintaining performance after pretraining for twice as long as initial convergence.*

3.2. Results

Overall Results. Table 1 shows the results of the MAE-AST with different encoder depths against the SSAST with equivalent input processing and masking techniques. We find that the MAE-AST matches or outperforms the SSAST on all downstream tasks but SID. We fail to make the 12-Layer Patch-Based MAE-AST converge on SID, and further discuss SID later.

Impact of the Number of Decoder Layers. In table 2, we show the downstream results from pretraining the default MAE-AST model with four different decoder depths, all other factors held constant. Like the MAE paper demonstrated for the visual domain, we find that increasing the decoder depth only provides small improvements if any, meaning that the decoder depth can be small relative to the encoder. This result is key in justifying the MAE-AST as an expandable architecture, as decoder layers are significantly more expensive than encoder layers. Interestingly, different tasks show different trends relative to the number of decoder layers, with AS and SID improving with fewer decoder layers and ER improving with more.

Impact of Masking Strategies. In table 3 we show results of eight main strategies of tokenizing and masking. Like the SSAST paper, we see that the most important factor for training is input tokenization, finding patch-based models better for audio tasks, and frame-based models better for speech tasks. We also observe that chunking for a patch-based MAE-AST improves downstream performance across the board, while this did not occur for the vision-based MAE. This could imply that our audio features are more correlated than MAE's input images. Finally, we find that high masking ratios and chunking provide a consistent benefit for patch-based models, while for frame-based models the results are more varied.

Pretraining Speed improvements. In table 4, we evaluate the DNN time per epoch and CUDA memory usage for pretraining the MAE-AST with different numbers of encoder layers and masking ratios. We find that the default SSAST implementation[1] is significantly slower than our implementa-

[1]https://github.com/YuanGongND/ssast

Table 1: *Overall Results. We do not replicate SSAST Base Frame on AudioSet for fairness because the exact fine-tune parameters aren't published. Note that the numbers we report for the SSAST model differ from those reported by [8]. This is due to the fact that we train and test all models on our modified version of the AudioSet dataset.*

Model	Enc. Layers	Masking	AS	ESC	KS2	KS1	SID	ER
SSAST Base Patch	12	Chunked	28.6	88.8	98.0	96.0	64.3	59.6
SSAST Base Frame	12	Random	-	85.9	**98.1**	96.7	**80.8**	60.5
MAE-AST Patch	6	Chunked	28.3	88.6	97.4	95.0	37.6	58.7
MAE-AST Frame	6	Random	25.9	88.0	97.8	96.6	58.6	60.2
MAE-AST Patch	12	Chunked	**30.6**	**90.0**	97.9	95.8	-	59.8
MAE-AST Frame	12	Random	23.0	88.9	98.0	**97.3**	63.3	**62.1**

Table 2: *Decoder Layer Ablation. Based on the default model with solely the decoder depth changed during pretraining*

Layers	AS	ESC	KS2	KS1	SID	ER
1	**28.8**	87.8	97.3	**95.5**	**42.8**	57.8
2	28.3	88.6	**97.4**	95.0	37.6	58.7
3	27.8	**88.7**	**97.4**	94.5	38.7	58.1
4	27.6	88.1	97.2	94.4	32.7	**59.1**

Table 3: *Masking Strategies Ablation. Results based on the default model. Tests the pairwise combinations of tokenization type, the masking ratio (%p), and the masking strategy.*

Input	%p	Masking	AS	KS2	ER
Patch	50	Random	27.6	97.4	57.3
Patch	50	Chunk	27.8	97.4	58.2
Patch	75	Random	26.7	97.4	58.6
Patch	75	Chunk	**28.3**	97.4	58.7
Frame	50	Random	23.6	97.8	59.1
Frame	50	Chunk	24.9	**97.9**	**62.1**
Frame	75	Random	25.9	97.8	60.2
Frame	75	Chunk	23.4	**97.9**	61.8

tion using fairseq[2], and so we additionally compare against the MAE-AST with 0 encoder layers and $n = 6, 12$ decoder layers to better represent the speedup from incorporating MAE-style pretraining. We denote this training with mask tokens included at every layer as "w/ [M]". MAE-AST achieves a $3\times$ speedup over pretraining with mask tokens included and a $2\times$ memory reduction, showing a significant and practical improvement in training ordinarily-sized models with modern hardware and efficient libraries. With MAE-AST also converging in the same number of iterations as SSAST (figure 2), we see promising results that the MAE-AST is both efficient and expandable.

Impact of Pretext Task. Table 5 shows the downstream performance of SSAST and default MAE-AST when only using the generative or discriminative losses. We find that the MAE-AST is significantly better than the SSAST when using only a generative loss. Alongside weak results on the lower level SID task, we hypothesize this is because the information bottleneck in MAE-AST conditions the model to further extract abstract data, similar to embedding spaces in standard Autoencoders. Our results also suggest that the discriminative loss is more useful for audio tasks while the generative loss is more useful for speech tasks.

[2]https://github.com/pytorch/fairseq

Table 4: *Pretraining Speed Ablation. Speed performance is tested for all models using a batch size of 32 and input audio length truncated to 10 seconds on a single Quadro RTX 8000. MAE-AST models use 2 decoder layers, which show high-quality results in table 2 and maximal fine-tune performance in the MAE paper [12]. We reimplement SSAST-like training with a decoder-only MAE-AST (w/ [M]) to account for significant library differences and highlight speed differences solely from the switch to an encoder-decoder architecture.*

Mask	Lay.	Sec/Ep	Mem. (MiB)	Speedup
75%	6	7016	6291	1.80×
75%	12	8299	8227	**2.96**×
50%	12	12222	11639	2.01×
w/ [M]	6	12636	9871	-
w/ [M]	12	24577	17719	-
SSAST [8]	12	150755	41547	-

Table 5: *Pretext Task Ablation. Evaluates a Generative only or Discriminative only loss. We compare the default patch-based SSAST to the default MAE-AST with retrained AudioSet*

Model	AS	ESC	KS2	KS1	SID	ER
SSAST Gen	12.2	74.2	96.6	93.3	40.1	54.3
SSAST Disc	**29.6**	**85.6**	**98.0**	**94.2**	**61.4**	**57.5**
Ours Gen	26.1	86.3	**97.3**	**95.1**	**40.6**	**58.6**
Ours Disc	**28.5**	**89.0**	**97.3**	94.6	38.8	58.3

4. Conclusion

We present the MAE-AST, a new model for self-supervised pretraining in the audio domain. We find our model significantly speeds up pretraining, reduces memory costs, and matches or beats the SSAST on the majority of downstream tasks. Like SSAST, we continue to observe that patch-based tokenization is better for audio tasks and frame-based tokenization is better for speech tasks. Furthermore, we find that MAE-like pretraining is more robust to different loss functions, greatly outperforming the SSAST when only using a generative task, but is significantly worse at low-level tasks like speaker identification. We hypothesize that this implies the MAE architecture is encouraging additional abstraction. We hope that future work can build on these findings toward the goal of constructing a scalable multi-domain strategy for self-supervised pretraining.

5. References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[4] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[5] W.-N. Hsu, Y.-H. H. Tsai, B. Bolte, R. Salakhutdinov, and A. Mohamed, "Hubert: How much can a bad teacher benefit asr pre-training?" in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6533–6537.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[7] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

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

[9] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," in *Interspeech*, 2021.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[11] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ArXiv*, vol. abs/2009.06732, 2020.

[12] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," *arXiv:2111.06377*, 2021.

[13] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[15] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[16] J. F. Gemmeke, D. P. 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.

[17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *ICASSP*, 2015.

[18] K. J. Piczak, "ESC: Dataset for environmental sound classification," in *Multimedia*, 2015.

[19] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[20] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, p. 101027, 2020.

[21] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "Iemocap: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, no. 4, pp. 335–359, 2008.

[22] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin *et al.*, "Superb: Speech processing universal performance benchmark," *arXiv preprint arXiv:2105.01051*, 2021.