FROM COARSE TO FINE:
EFFICIENT TRAINING FOR AUDIO SPECTROGRAM TRANSFORMERS

Jiu Feng, Mehmet Hamza Erol*, Joon Son Chung, Arda Senocak*

Korea Advanced Institute of Science and Technology, South Korea

ABSTRACT

Transformers have become central to recent advances in audio classification. However, training an audio spectrogram transformer, *e.g.* AST, from scratch can be resource and time-intensive. Furthermore, the complexity of transformers heavily depends on the input audio spectrogram size. In this work, we aim to optimize AST training by linking to the resolution in the time-axis. We introduce multi-phase training of audio spectrogram transformers by connecting the seminal idea of coarse-to-fine with transformer models. To achieve this, we propose a set of methods for temporal compression. By employing one of these methods, the transformer model learns from lower-resolution (coarse) data in the initial phases, and then is fine-tuned with high-resolution data later in a curriculum learning strategy. Experimental results demonstrate that the proposed training mechanism for AST leads to improved (or on-par) performance with faster convergence, *i.e.* requiring fewer computational resources and less time. This approach is also generalizable to other AST-based methods regardless of their learning paradigms.

***Index Terms*—** Audio Spectrogram Transformers, Audio Classification, Efficient Training, Temporal Redundancy

1. INTRODUCTION

Convolutional Neural Networks (CNNs) and, more recently, transformers have made a significant impact on numerous computer vision and audio processing tasks. Among these tasks, audio classification is a central research topic that assigns labels to the given audio inputs. The existing transformer-based approaches employ a patch-based system for audio classification [1, 2, 3, 4, 5, 6, 7, 8], where the input spectrograms are divided into fixed-size patches to create tokens as input for the transformer backbone. With the paradigm shift toward transformer-based approaches, an emerging thread of work also aims to explore efficient ways of optimizing the complexity of transformers, as it increases quadratically with the input sequence length. Recent works aim to reduce the quadratic complexity to make transformers more efficient for audio processing applications. Koutini *et al.* [4] propose a method called Patchout, which efficiently drops patches while concurrently disentangling the positional encodings of both time and frequency axes. Later, masked auto-encoder approach is employed to reduce the number of tokens, either through reconstruction objectives [5, 6, 7] or direct prediction of representations [9] for masked input patches. Unlike the approach of dropping the input patches, HTS-AT [3] leverages a hierarchical shifted window approach known as the Swin Transformer [10], originally employed in the domain of vision. Subsequently, another hierarchical strategy, known as multi-scale transformers [8, 11], is applied in the audio domain by hierarchically expanding the channels while reducing the spatial resolution in the model.

Method	Architecture	Multi-Phase	AudioSet (mAP)
Baseline - CNN14 [12]	CNN	✗	43.20
Pool2 [13]	CNN	✗	42.60
Baseline - AST [1]	Transformer	✗	44.30
Pool2	Transformer	✗	42.79
Pool2→1	Transformer	✓	44.35

Table 1: **Audio transformer models need multi-phase training for efficient learning.** Simple low-resolution training only works for CNNs. “Pool2” indicates training on 2 times compressed spectrograms. “Pool2 → 1” denotes initial phase of training on 2 times compressed spectrograms, followed by training on full resolution.

As aforementioned, the existing transformer-based approaches in the audio domain take spectrograms as input. When extracting the spectrograms, varying resolutions in time result in a different number of tokens to train the transformer backbone. Our goal is to link the resolution in the time-axis to efficient training of Audio Spectrogram Transformers [1]. We posit that employing different resolutions in training audio classification models can be both intuitive and beneficial. Firstly, as discussed in [13, 14], spectrograms may exhibit temporal redundancy. Audio patterns can be uniformly continuous or periodic [15, 16]. Thus, shortening the time dimension can eliminate the redundancy in the input spectrogram, resulting in a reduction in computational cost and time. Secondly, following a fundamental principle in vision on input resolution, models that learn from coarse to fine-grained data can achieve performance improvements due to the scale invariance of the representations.

The use of reduced input signals for computational efficiency in audio classification has been investigated by [17, 13] in the context of CNNs only. Xubo *et al.* [13] propose simple pooling methods, such as max pooling, average pooling, and *etc.*, to eliminate temporally redundant information and enhance efficiency. By training the model with temporally reduced audio input, it performs similarly to the baseline that uses the original input, as shown in Table 1. However, training the audio spectrogram transformer (AST) directly with the temporally reduced input results in a significant performance drop compared to the baseline. This highlights that training efficient ASTs with lower-resolution inputs requires a different approach. We conjecture that training ASTs in a curriculum learning fashion, starting from coarse to fine-grained data, can help mitigate this issue [18, 19]. As shown in Table 1, curriculum learning (fine-tuning with higher resolution data) achieves comparable performance while also reducing computational costs and time.

In this work, we propose curriculum learning-based training using the resolution of the audio signals as a proxy. This approach leads to efficient audio spectrogram transformer training, achieving comparable or better accuracy-to-computation tradeoffs compared to

*These authors contributed equally to this work.

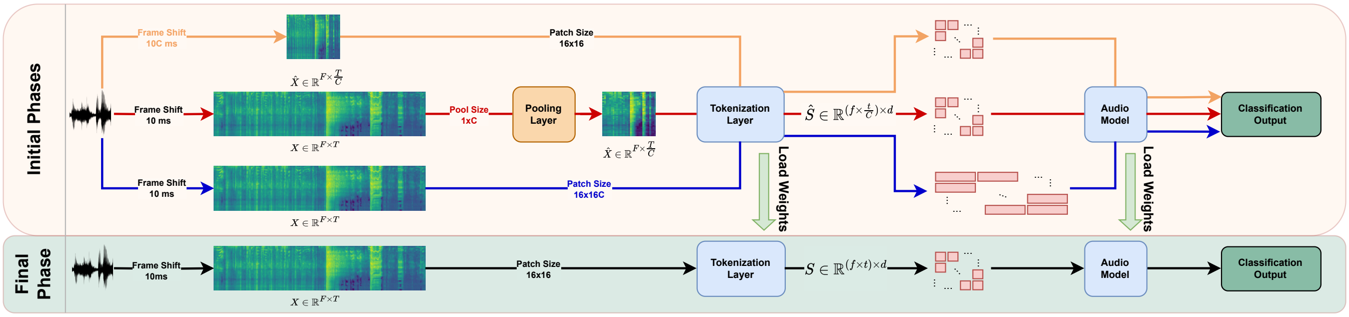

Fig. 1: **Illustration of initial and final phase pipelines in our proposed training method.** Fshift, Pool, and Patch are compression methods from Section 2.3. In the initial training phases, only one of them will be employed to get $f \times \frac{t}{C}$ number of tokens. Each method's unique contribution compared to the original pipeline is color-highlighted. Given numbers reflect the AST's original training settings.

the widely used AST. To accomplish this task, we take the following steps (shown in Figure 1): (1) AST training is split into two (or multiple) phases. In the initial phase, the model is trained with input that has a lower resolution in the time-axis obtained through various reduction methods. Subsequent phases leverage higher resolution (eventually the original resolution) audio data. (2) We design various reduction methods, such as Frame-Shift, Pooling, and Patchification. One of these methods is employed during the initial phases of training. (3) While transitioning into subsequent phases, the model weights (*e.g*. positional embeddings) are appropriately adapted to match the token numbers of the new phase. Based on these steps, our proposed training mechanism for AST [1] yields improved (or on-par) performance and requires fewer resources compared to the baseline model on four standard audio classification datasets. We conduct extensive ablation studies of our design choices. Moreover, we demonstrate that this approach can be further generalized to other AST-based approaches, such as HTS-AT [3] and SSAST [2].

2. APPROACH

2.1. Mel-Spectrogram and Complexity

Given a waveform $W \in \mathbb{R}^{1\times L}$, an Audio Spectrogram Transformer processes its corresponding mel-spectrogram (mel-spec), represented as $X = mel(W) \in \mathbb{R}^{F\times T}$, where $mel(\cdot)$ denotes the spectrogram generator module. This mel-spectrogram is first patchified and then tokenized into a sequence of tokens $S = Token(X) \in \mathbb{R}^{(f\times t)\times d}$, using the tokenization layer $Token(\cdot)$. The term $f \times t$ indicates the number of tokens. This sequence of tokens then serves as the input for the transformer's encoder layers. It is important to note that when using square-shaped patches, the length of the time axis has a significant impact on the token count. This, in turn, influences the complexity, which grows quadratically. Drawing from the insights of [13], we hypothesize that mel-specs may contain surplus temporal information during the early training phases, as the model might not need such a detailed representation initially. Therefore, by starting the training with a coarser temporal perspective and refining it progressively, we can improve the learning efficiency by potentially achieving comparable or better accuracy-to-computation tradeoffs compared to the traditional training methods.

2.2. Multi-phase Training

Our method splits the training into multiple phases. In the first phase, we apply one of the temporal compression methods (details are in Section 2.3) to the input mel-specs, reducing the number of tokens along the time axis by a factor of C, yielding $f \times \frac{t}{C}$ tokens. By introducing coarse data, the model can quickly assimilate generalized features, which guides the model weights into a good latent space. During the transition into the subsequent phases, the value of C is reduced, and we transfer the trained weights from the previous phase by appropriately adapting the parameters that depend on the number of tokens (*e.g*. interpolating the positional embeddings). Furthermore, the training settings, such as the learning rate, optimizer, and *etc*., are reset to their initial values.

2.3. Compression Methods

We investigate three distinct strategies for the temporal compression. Figure 1 provides an illustration depicting the roles of these methods.
Change Frame-Shift Size (Fshift): Mel-spectrograms are constructed by specifying frame-size and frame-shift values, which are, for instance, set to 25ms and 10ms by default in AST [1]. In this method, the frame-shift value is multiplied by a factor of C when generating a mel-spec, which results in a temporally compressed mel-spec $\hat{X}$.

$$\hat{X} = mel'(W) \in \mathbb{R}^{F\times\frac{T}{C}} \tag{1}$$

Max/Avg Pooling (Pool): Before tokenizing a mel-spec, we pass it through an additional max- or average-pooling layer with the kernel and stride of size $1 \times C$, resulting in a temporal reduction of the mel-spec by a factor of C.

$$\begin{aligned} \text{Avg(X):}\quad \hat{X}[i,j] &= \frac{1}{C}\sum_{n=0}^{C-1} X[i, C\cdot j + n] \\ \text{Max(X):}\quad \hat{X}[i,j] &= \max_{n\in[0,C-1]} X[i, C\cdot j + n] \end{aligned} \tag{2}$$

where $i \in [0, F)$ and $j \in [0, \frac{T}{C})$.

Flexible Patchification (Patch): In the tokenization process, patches typically have a square shape, denoted as $p \times p$. Inspired by [20], we apply a rectangular patch size of $p \times Cp$ during tokenization. As each patch becomes C times wider, the number of patches along the time dimension also decreases by a factor of C, resulting in temporal compression. Note that even though this method uses the full-resolution audio signal, we still consider it a compression method since it reduces the number of tokens. The models that employ this method apply either bilinear interpolation (BL) or PI-Resize operator (PI) [20, 21] when transferring the patch embedding weights from the previous phase.

$$\hat{S} = Token'(X) \in \mathbb{R}^{(f\times\frac{t}{C})\times d} \tag{3}$$

Setting	VGGSound			VoxCeleb			Kinetics-Sounds			AudioSet		
	Acc	FLOPs Save(%)	Time Save(%)	Acc	FLOPs Save(%)	Time Save(%)	Acc	FLOPs Save(%)	Time Save(%)	mAP	FLOPs Save(%)	Time Save(%)
Baseline	49.40	-	-	41.90	-	-	62.92	-	-	44.30	-	-
Fshift 4→1	49.90 (+0.50)	58.31	55.00	41.95 (+0.05)	42.56	38.89	63.84 (+0.92)	42.56	38.89	43.82 (-0.48)	35.32	33.40
Fshift 2→1	51.43 (+2.03)	46.16	45.00	43.68 (+1.78)	40.18	38.89	63.22 (+0.30)	40.18	38.89	44.10 (-0.21)	30.47	30.00
Avg Pool 4→1	50.49 (+1.09)	58.31	55.00	42.20 (+0.30)	31.45	27.78	62.99 (+0.07)	31.45	27.78	43.98 (-0.32)	35.32	33.40
Max Pool 4→1	50.21 (+0.81)	58.31	55.00	42.01 (+0.11)	53.67	50.00	63.14 (+0.22)	31.45	28.00	43.87 (-0.43)	35.32	33.40
Avg Pool 2→1	49.85 (+0.45)	46.16	45.00	43.75 (+1.85)	40.18	38.89	63.14 (+0.22)	29.07	27.78	43.98 (-0.32)	30.47	30.00
Max Pool 2→1	49.66 (+0.26)	46.16	45.00	43.95 (+2.05)	40.18	38.89	63.66 (+0.74)	17.96	16.67	44.19 (-0.11)	30.47	30.00
Patch BL 4→1	50.61 (+1.21)	58.23	55.00	41.49 (-0.41)	31.36	27.78	63.84 (+0.92)	31.36	27.78	43.96 (-0.34)	35.29	33.40
Patch PI 4→1	50.44 (+1.04)	58.23	55.00	42.07 (+0.17)	20.25	16.67	63.22 (+0.30)	31.36	27.78	44.05 (-0.25)	35.29	33.40
Patch BL 2→1	51.18 (+1.78)	46.11	45.00	44.65 (+2.75)	40.12	38.89	63.33 (+0.41)	29.01	27.78	44.14 (-0.16)	30.44	30.00
Patch PI 2→1	51.61 (+2.21)	46.11	45.00	45.09 (+3.19)	40.12	38.89	63.18 (+0.26)	29.01	27.78	44.16 (-0.14)	30.44	30.00

Table 2: **FLOPs and time-saving ratios from 2-phase experiments using proposed compression methods.**

Setting	VGGSound (Acc)	VoxCeleb (Acc)	Kinetics-Sounds (Acc)	AudioSet (mAP)
Baseline	49.40	41.90	62.92	44.30
Fshift 4→1	52.31 (+ 2.91)	43.22 (+1.32)	64.14 (+1.22)	44.17 (-0.13)
Fshift 2→1	52.93 (+3.53)	46.71 (+4.81)	64.73 (+1.81)	44.30 (=)
Avg Pool 4→1	52.76 (+3.36)	43.13 (+1.24)	63.81 (+0.89)	44.28 (-0.02)
Max Pool 4→1	52.45 (+3.05)	44.69 (+2.79)	63.62 (+0.70)	44.19 (-0.11)
Avg Pool 2→1	53.42 (+4.02)	46.47 (+4.57)	64.21 (+1.29)	44.22 (-0.08)
Max Pool 2→1	53.17 (+3.77)	46.64 (+4.74)	64.03 (+1.11)	44.35 (+0.05)
Patch BL 4→1	52.97 (+3.57)	41.89 (-0.01)	63.84 (+0.92)	44.17 (-0.13)
Patch PI 4→1	52.48 (+3.08)	42.07 (+0.17)	63.73 (+0.81)	44.28 (-0.02)
Patch BL 2→1	53.08 (+3.68)	46.64 (+4.74)	64.33 (+1.41)	44.38 (+0.08)
Patch PI 2→1	53.17 (+3.77)	47.63 (+5.73)	64.18 (+1.26)	44.38 (+0.08)

Table 3: **Performance in the 2-phase approach when trained until convergence without training budget constraints.**

3. EXPERIMENTS

3.1. Datasets and Evaluation Metrics

Datasets. We conduct experiments using four datasets: (1) AudioSet, (2) VGGSound, (3) VoxCeleb, and (4) Kinetics-Sounds. AudioSet [22] is a large-scale multi-label dataset with approximately 2 million 10-second clips, featuring 527 labels across diverse audio categories. The balanced set is curated from the full set by selecting around 20K samples. VGGSound [23] consists of ~200K 10-second videos labeled with 309 sound classes. VoxCeleb [24] provides an audio-visual dataset of human speech, containing 1251 speakers with approximately 145,000 utterances. Kinetics-Sounds is a subset of Kinetics [25], constructed from 10-second audio clips from YouTube. In our case, we use around 20K and 2.7K audio samples for training and testing, respectively.

Evaluation metrics. Due to the existence of multiple labels in each sample of AudioSet, we use mean average precision (mAP) across all classes for the evaluation. For the other datasets, we report the Top-1 classification accuracy (Acc) as samples are assigned only a single label.

3.2. Experiment setup

Implementation details of baselines. We train the AST baseline on AudioSet for 5 epochs by using the official configurations in [1]. For VGGSound, VoxCeleb, and Kinetics-Sounds, we train for 20 epochs and adopt the setup of the AudioSet training. However, mixup augmentation and weighted averaging are not utilized in these three datasets. To expedite the processing, non-overlapping patches are applied to all the datasets. Note that, unlike [2, 5], our VoxCeleb experiments follow the same training and evaluation pipeline as the other three datasets, instead of the SUPERB framework [26], to maintain consistency in implementation and experiments. For the HTS-AT baseline on AudioSet, we adopt the full setting from [3] but train for 25 epochs and report the weighted averaging result of the top 15 checkpoints. On VGGSound, we train for 50 epochs but omit the weighted averaging result. Lastly in SSAST, we follow the settings of [2] and perform self-supervised pretraining with patch-based model for 800k iterations on joint AudioSet and LibriSpeech [27] datasets. The setups of supervised fine-tuning are identical to what we use in AST. More details are available at https://sites.google.com/view/coarse-to-fine-audio.

3.3. Main Results

This section presents the results of our proposed training mechanism in terms of accuracy and resource efficiency, where training is split into two phases. We temporally compress the mel-spectrograms by a factor of C during the initial phase, and we report the results for C values of 2 and 4. The initial training phase is set to approximately 25% of the total number of training epochs in the baseline settings. Following this rule, our model is trained for 1 epoch on AudioSet and 5 epochs for the remaining datasets in the initial phase. Afterwards, we transfer the trained weights as the initialization of the final phase, where we use high-resolution (*i.e.* original resolution) data for fine-tuning. When loading weights for the second training phase, we resize the positional embedding dimensions through bilinear interpolation to accommodate the change in the number of tokens.

Accuracy/Computation trade-offs. We analyze our model from the perspective of saving computational resources. To achieve this, we terminate the final phase of training at the earliest epoch that surpasses the baseline performance. All the differences in computational savings are calculated based on the baseline reference, where the epoch number with the highest accuracy is selected. The results presented in Table 2 show that we save from 18% to 58% of FLOPS while maintaining on-par or better performance than the baseline on different datasets. The only exception to this is AudioSet, where we save more than 30% of FLOPs with a negligible drop in mAP as in [13]. Another observation is that there is no obvious difference between the compression methods. This highlights that the coarse-to-fine approach with time-axis compression is beneficial for the efficient training of AST, regardless of the compression method.

Accuracy/No computational budget constraints. In contrast to the previous analysis, here we allow the model to train until convergence without considering training budget constraints, solely aiming to achieve the highest performance improvement. As displayed in Table 3, AST consistently achieves further performance improve-

Setting	VGGSound			VoxCeleb			Kinetics-Sounds		
	Acc	FLOPs Save (%)	Time Save (%)	Acc	FLOPs Save (%)	Time Save (%)	Acc	FLOPs Save (%)	Time Save (%)
Baseline	49.40	-	-	41.90	-	-	62.92	-	-
Fshift 4→2→1	49.97 (+0.57)	58.68	56.00	42.83 (+0.93)	31.87	28.89	64.36 (+1.44)	42.98	40.00
Avg Pool 4→2→1	50.54 (+1.14)	58.68	56.00	42.42 (+0.52)	42.98	40.00	63.44 (+0.52)	42.98	40.00
Max Pool 4→2→1	50.76 (+1.36)	58.68	56.00	42.29 (+0.39)	42.98	40.00	64.07 (+1.15)	31.87	28.89
Patch BL 4→2→1	50.09 (+0.69)	58.60	56.00	41.29 (-0.61)	20.67	17.78	63.99 (+1.07)	42.89	40.00
Patch PI 4→2→1	50.24 (+0.84)	58.60	56.00	42.25 (+0.35)	31.78	28.89	63.99 (+1.07)	42.89	40.00

Table 4: **FLOPs and time-saving ratios from 3-phase experiments using proposed compression methods.**

Setting	VGGSound (Acc)	VoxCeleb (Acc)	Kinetics-Sounds (Acc)
Baseline	49.40	41.90	62.92
Fshift 4→2→1	53.51 (+4.11)	43.81 (+1.91)	65.25 (+2.33)
Avg Pool 4→2→1	53.52 (+4.12)	43.91 (+2.01)	65.25 (+2.33)
Max Pool 4→2→1	53.48 (+4.08)	44.41 (+2.51)	64.58 (+1.66)
Patch BL 4→2→1	53.80 (+4.40)	41.89 (-0.01)	64.99 (+2.07)
Patch PI 4→2→1	53.62 (+4.22)	43.11 (+1.21)	65.43 (+2.51)

Table 5: **Performance in the 3-phase approach when trained until convergence.**

Setting	FLOPs Save (%)	AudioSet		VGGSound	
		mAP	Time Save (%)	Acc	Time Save (%)
Baseline HTS-AT	-	46.92	-	52.82	-
Fshift 4→1	21.13	46.75 (-0.17)	10.50	52.98 (+0.16)	14.32
Baseline SSAST	-	29.42	-	45.46	-
Fshift 4→1	34.58	30.28 (+0.87)	31.73	47.78 (+2.31)	31.73

Table 6: **Results for HTS-AT (using AudioSet-2M) and SSAST (using AudioSet-20K).** Experiments utilize the Fshift compression method in 2 phases, beginning with $C = 4$.

ments, up to a 4% improvement on VGGSound, except for AudioSet where the improvements are negligible. These results demonstrate that the coarse-to-fine approach enables the model to begin learning from high-level information and gradually progress to important details, thus leading to better performance.

3.4. Ablation on Multi-Phases of Higher Resolution Fine-Tuning

In the main experiments, the model is trained with one low-resolution phase and one high-resolution fine-tuning phase. In this section, we further study the impact of using multiple phases for fine-tuning. Here, an additional phase is inserted between the initial phase and the final phase of fine-tuning, resulting in a three-phase training. Specifically, the model is sequentially trained with the (4→2→1) variant, where the model is first trained with a compression rate of $C = 4$, and then progressively fine-tuned with $C = 2$ and $C = 1$, which represents the original resolution in the final phase. We schedule the intervals for the initial phases as 30% of the baseline total training, resulting in 3 epochs for the initial phases.

Similar to the main experiments, the termination of the final fine-tuning phase is decided based on two criteria: (1) the earliest epoch that surpasses the baseline performance, (2) training until convergence. While the first criterion is for exploring the Accuracy/Computation trade-offs perspective, the latter one is used to achieve the highest performance improvement without considering computational resource constraints. The results are shown in Table 4 and Table 5. As the results demonstrate, three-phase training also brings both training efficiency and performance improvements compared to the baseline. However, we observe that two-phase training provides competitive performance to three-phase training. Therefore, we use two-phase training for simplicity.

3.5. Generalization on Different Baselines

To demonstrate the general applicability of the coarse-to-fine training approach with time resolution reduction to other methods, we conduct experiments with recent methods, SSAST [2] and HTS-AT [3], by simply applying our proposed training mechanism to them. All of these baselines are audio spectrogram transformer (AST) based methods. For simplicity, we employ the Fshift compression method with the two-phase approach (4→1) and the final fine-tuning phase is terminated when it surpasses the baseline performance. The models are evaluated on the VGGSound and AudioSet datasets in these experiments. The results are shown in Table 6.

HTS-AT. Similar to the main experiments, here we set the duration of the initial phase training to approximately 25% of the total number of training epochs in the HTS-AT baseline settings. As Table 6 illustrates, the coarse-to-fine training approach saves around 20% of training FLOPs while providing on-par performance with the baseline. Note that HTS-AT is already a very efficient transformer-based method, and our training paradigm further enhances its efficiency.

SSAST. Since SSAST is a self-supervised method, it undergoes pre-training before being applied to downstream tasks [2]. Generally, the pre-training phase is the most time and computation-intensive stage. Therefore, we apply our coarse-to-fine training paradigm during the pre-training stage. The model is initially pre-trained for 100K iterations with low-resolution data, followed by 500K iterations in the final phase with the original resolution. We utilize the joint AudioSet and LibriSpeech datasets for training, adhering to the baseline setting. As shown in Table 6, the coarse-to-fine training paradigm leads to improved performance in downstream tasks, while simultaneously achieving more than a 30% reduction in training FLOPs and time. Moreover, these results indicate that the proposed training mechanism is generalizable to other AST-based methods, regardless of their learning paradigms, *i.e.* whether supervised or self-supervised.

4. CONCLUSION

In this paper, we focus on the efficient training of audio spectrogram transformers with the motivation of temporal redundancy in spectrograms. We propose a coarse-to-fine training, initially using low-resolution input in the time-axis, progressively fine-tuning with higher resolution. Our experiments demonstrate on-par or better performance while saving computational resources. Furthermore, we show that this approach is generalizable to other AST-based methods. Transformers achieve optimal performance with large datasets [28]. Considering that the audio domain does not have a dataset of similar size compared to vision yet, efficient training for audio transformers will play a crucial role in future research. Moreover, learnable schedulers for the phase transition can be also explored.

5. REFERENCES

[1] Yuan Gong, Yu-An Chung, and James Glass, "AST: audio spectrogram transformer," in *Proc. Interspeech*, 2021.

[2] Yuan Gong, Cheng-I Lai, Yu-An Chung, and James Glass, "SSAST: self-supervised audio spectrogram transformer," in *Proc. AAAI*, 2022.

[3] Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "HTS-AT: a hierarchical token-semantic audio transformer for sound classification and detection," in *Proc. ICASSP*, 2022.

[4] Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer, "Efficient training of audio transformers with patchout," in *Proc. Interspeech*, 2022.

[5] Alan Baade, Puyuan Peng, and David Harwath, "MAE-AST: masked autoencoding audio spectrogram transformer," in *Proc. Interspeech*, 2022.

[6] Po-Yao Huang, Hu Xu, Juncheng B Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, and Christoph Feichtenhofer, "Masked autoencoders that listen," in *NeurIPS*, 2022.

[7] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino, "Masked spectrogram modeling using masked autoencoders for learning general-purpose audio representation," in *PMLR*, 2022.

[8] Wentao Zhu and Mohamed Omar, "Multiscale audio spectrogram transformer for efficient audio classification," in *Proc. ICASSP*, 2023.

[9] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino, "Masked modeling duo: Learning representations by encouraging both networks to model the input," in *Proc. ICASSP*, 2023.

[10] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. CVPR*, 2021.

[11] Yuchen Liu, Natasha Ong, Kaiyan Peng, Bo Xiong, Qifan Wang, Rui Hou, Madian Khabsa, Kaiyue Yang, David Liu, Donald S Williamson, et al., "Mmvit: Multiscale multiview vision transformers," *arXiv preprint arXiv:2305.00104*, 2023.

[12] 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/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[13] Xubo Liu, Haohe Liu, Qiuqiang Kong, Xinhao Mei, Mark D. Plumbley, and Wenwu Wang, "Simple pooling front-ends for efficient audio classification," in *Proc. ICASSP*, 2023.

[14] Haohe Liu, Xubo Liu, Qiuqiang Kong, Wenwu Wang, and Mark D Plumbley, "Learning the spectrogram temporal resolution for audio classification," *arXiv preprint arXiv:2210.01719*, 2022.

[15] Honglie Chen, Weidi Xie, Triantafyllos Afouras, Arsha Nagrani, Andrea Vedaldi, and Andrew Zisserman, "Audio-visual synchronisation in the wild," *arXiv preprint arXiv:2112.04432*, 2021.

[16] Arda Senocak, Junsik Kim, Tae-Hyun Oh, Dingzeyu Li, and In So Kweon, "Event-specific audio-visual fusion layers: A simple and new perspective on video understanding," in *Proc. WACV*, 2023.

[17] Federico Colangelo, Federica Battisti, and Alessandro Neri, "Progressive training of convolutional neural networks for acoustic events classification," in *EUSIPCO*, 2021.

[18] Saghar Irandoust, Thibaut Durand, Yunduz Rakhmangulova, Wenjie Zi, and Hossein Hajimirsadeghi, "Training a vision transformer from scratch in less than 24 hours with 1 gpu," *arXiv preprint arXiv:2211.05187*, 2022.

[19] Runze Li, Dahun Kim, Bir Bhanu, and Weicheng Kuo, "Reclip: Resource-efficient clip by training with small images," *arXiv preprint arXiv:2304.06028*, 2023.

[20] Jiu Feng, Mehmet Hamza Erol, Joon Son Chung, and Arda Senocak, "FlexiAST: Flexibility is what AST needs," in *Proc. Interspeech*, 2023.

[21] Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic, "FlexiViT: One model for all patch sizes," in *Proc. ICCV*, 2023.

[22] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter, "AudioSet: An ontology and human-labeled dataset for audio events," in *Proc. ICASSP*, 2017.

[23] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman, "VGGSound: A large-scale audio-visual dataset," in *Proc. ICASSP*, 2020.

[24] Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman, "VoxCeleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, 2020.

[25] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al., "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[26] Shu-Wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y. Lin, Andy T. Liu, Jiatong Shi, Xuankai Chang, et al., "SUPERB: speech processing universal performance benchmark," in *Proc. Interspeech*, 2021.

[27] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015.

[28] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. ICLR*, 2021.