HTS-AT: A HIERARCHICAL TOKEN-SEMANTIC AUDIO TRANSFORMER FOR SOUND CLASSIFICATION AND DETECTION

Ke Chen [1], Xingjian Du [2], Bilei Zhu [2], Zejun Ma [2], Taylor Berg-Kirkpatrick [1], Shlomo Dubnov [1]

[1]University of California San Diego   [2]AI Lab, Bytedance Inc.

ABSTRACT

Audio classification is an important task of mapping audio samples into their corresponding labels. Recently, the transformer model with self-attention mechanisms has been adopted in this field. However, existing audio transformers require large GPU memories and long training time, meanwhile relying on pretrained vision models to achieve high performance, which limits the model's scalability in audio tasks. To combat these problems, we introduce HTS-AT: an audio transformer with a hierarchical structure to reduce the model size and training time. It is further combined with a token-semantic module to map final outputs into class featuremaps, thus enabling the model for the audio event detection (i.e. localization in time). We evaluate HTS-AT on three datasets of audio classification where it achieves new state-of-the-art (SOTA) results on AudioSet and ESC-50, and equals the SOTA on Speech Command V2. It also achieves better performance in event localization than the previous CNN-based models. Moreover, HTS-AT requires only 35% model parameters and 15% training time of the previous audio transformer. These results demonstrate the high performance and high efficiency of HTS-AT.

Index Terms— Audio Classification, Sound Event Detection, Transformer, Token-Semantic Module

1. INTRODUCTION

Audio classification is an audio retrieval task which aims to learn a mapping from audio samples to their corresponding labels. Depending on the audio categories, it involves sound event detection [1], music instrument classification [2], among others. It establishes a foundation for many downstream applications including music recommendation [3], keyword spotting [4], music generation [5, 6], etc.

With burgeoning research in the field of artificial intelligence, we have seen significant promising progress in audio classification. For data collections, many datasets with different types of audio (e.g. AudioSet [7], ESC-50 [8], Speech Command [4], etc.) provide platforms for the training and evaluation of models on different subtasks. For

the model design, the audio classification task is thriving based on neural-network-based models. Convolutional neural networks (CNNs) have been widely used in this field, such as DeepResNet [9], TALNet [10], PANN [11], and PSLA [12]. These models leverage CNN to capture features on the audio spectrogram, and further improve their performance through the design of the depth and breadth of the network. Recently, by introducing the transformer structure [13] into audio classification, the audio spectrogram transformer (AST) [14] further achieves the best performance through the self-attention mechanism and the pretrained model from computer vision. In this paper, we take a further step on a transformer-based audio classification model by first analyzing remaining problems in the AST.

First, since the transformer takes the audio spectrogram as a complete sequential data, AST takes a long time to train and consumes large GPU memories. In practice, it takes about one week to train on the full AudioSet with four 12GB GPUs. One method to boost training speed is to use the ImageNet [15] pretrained model in computer vision. However, this also limits the model to those pretrained hyperparameters, which reduces its scalability in more audio tasks. Indeed, we find that without pretraining, AST can only achieve the baseline performance (mAP=0.366 on AudioSet), which raises our attention to its learning efficiency on the audio data. Second, AST uses a class-token (CLS) to predict labels, making it unable to predict the start and end time of events in audio samples. Most CNN-based models naturally support the frame-level localization by empirically taking the penultimate layer's output as a event presence map. This inspires us to design a module that makes every output token of an audio transformer aware of the semantic meaning of events (i.e. a token-semantic module [16]) for supporting more audio tasks (e.g. sound event detection and localization).

In this paper, we propose HTS-AT[1], a hierarchical audio transformer with a token-semantic module for audio classification. Our contributions of HTS-AT can be listed as:

- HTS-AT achieves or equals SOTAs on AudioSet and ESC-50, and Speech Command V2 datasets. Moreover, the model without pretraining can still achieve the performance that is only 1%-2% lower than the best results.

---

[1]https://github.com/RetroCirce/HTS-Audio-Transformer
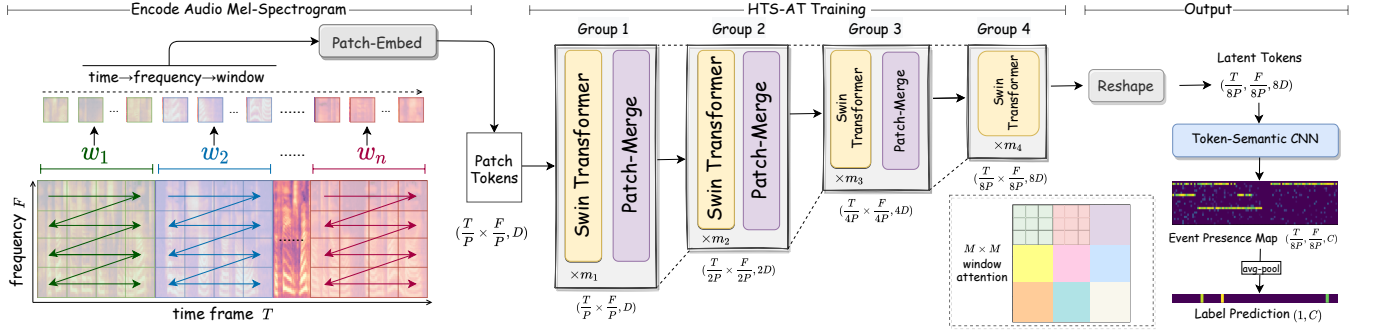
ICASSP 2022

Fig. 1: The model architecture of HTS-AT.

- HTS-AT takes fewer parameters (31M vs. 87M), fewer GPU memories, and less training time (80 hrs vs. 600 hrs) than AST's to achieve the best performance.

- HTS-AT further enables the audio transformer to produce the localization results of event only with weakly-labeled data. And it achieves a better performance than the previous CNN-based model.

2. PROPOSED MODEL

2.1. Hierarchical Transformer with Window Attention

A typical transformer structure consumes lots of GPU memories and training time, because the length of input tokens is too long and remains unchanged in all transformer blocks from beginning to end. As a result, the machine saves the output and its gradient of each block via large GPU memories, and spends much calculation time maintaining a large global self-attention matrix. To combat these problems, as depicted in Figure 1, we propose two key designs: a hierarchical transformer structure and a window attention mechanism.

2.1.1. Encode the Audio Spectrogram

In the left of Figure 1, an audio mel-spectrogram is cut into different patch tokens with a Patch-Embed CNN of kernel size $(P \times P)$ and sent into the transformer in order. Different from images, the width and the height of an audio mel-spectrogram denote different information (i.e. the time and the frequency bin). And the length of time is usually much longer than that of frequency bins. Therefore, to better capture the relationship among frequency bins of the same time frame, we first split the mel-spectrogram into **patch windows** $w_1, w_2, ..., w_n$ and then split the patches inside each window. The order of tokens follows **time→frequency→window** as shown in Figure 1. With this order, patches with different frequency bins at the same time frame will be organized adjacently in the input sequence.

2.1.2. Patch-Merge and Window Attention

In the middle of Figure 1, the patch tokens are sent into several groups of transformer-encoder blocks. At the end of each group, we implement a Patch-Merge layer [17] to reduce the

sequence size. This merge operation is applied by first reshaping the sequence to its original 2D map $(\frac{T}{P} \times \frac{F}{P}, D)$, where D is the latent state dimension. Then it merges adjacent patches as $(\frac{T}{2P} \times \frac{F}{2P}, 4D)$ and finally applies a linear layer to reduce the latent dimension to $(\frac{T}{2P} \times \frac{F}{2P}, 2D)$. As illustrated in Figure 1, the shape of the patch tokens is reduced by 8 times from $(\frac{T}{P} \times \frac{F}{P}, D)$ to $(\frac{T}{8P} \times \frac{F}{8P}, 8D)$ after 4 network groups, thus the GPU memory consumption is reduced exponentially after each group.

For each transformer block inside the group, we adopt a window attention mechanism to reduce the calculation. As shown in different color boxes in the middle right of Figure 1, we first split the patch tokens (in 2D format) into non-overlapping $(M \times M)$ **attention windows** $aw_1, aw_2, ..., aw_k$. Then we only compute the attention matrix inside each $M \times M$ attention window. As a result, we have k window attention (WA) matrices instead of a whole global attention (GA) matrix. The computational complexities of these two mechanisms in one transformer block for $f \times t$ audio patch tokens with the initial latent dimension D are:

$$\text{GA:} \quad \mathcal{O}(ftD^2 + (ft)^2 D) \tag{1}$$

$$\text{WA:} \quad \mathcal{O}(ftD^2 + M^2 ftD) \tag{2}$$

where the window attention reduces the second complexity term by $(\frac{ft}{M^2})$ times. For audio patch tokens in a time-frequency-window order, each window attention module will calculate the relation in a certain range of continuous frequency bins and time frames. As the network goes deeper, the Patch-Merge layer will merge adjacent windows, thus the attention relation is calculated in a larger space. In the code implementation, we use the swin transformer block with a **shifted** window attention [17], a more efficient window attention mechanism. This also helps us to use the swin transformer pretrained vision model in the experiment stage.

2.2. Token Semantic Module

The existing AST uses a class-token (CLS) to predict the classification label, which limits it from further indicating the start and end times of events as realized in CNN-based models. In the final layer output, each token contains information about its corresponding time frames and frequency bins. We

Model	Pretrain	#Params.	mAP	Ensemble-mAP
Baseline [7]	✗	2.6M	0.314	-
DeepRes [9]	✗	26M	0.392	-
PANN [11]	✗	81M	0.434	-
PSLA^P [12]	✓	13.6M	0.444	0.474
AST [14]	✗	87M	0.366	-
AST^P [14]	✓	87M	0.459	0.475 (0.485[2])
HTS-AT^H	✗	28.8M	0.440	-
HTS-AT^{HC}	✗	31M	0.453	-
HTS-AT^{HCP}	✓	31M	**0.471**	**0.487**

Table 1: The mAP results on AudioSet evaluation set.

Model	ESC-50 Acc.(%)	Model	SCV2 Acc.(%)
PANN [11]	90.5	RES-15 [21]	97.0
AST [14]	95.6 ± 0.4	AST [14]	$\mathbf{98.1 \pm 0.05}$
ERANN [22]	96.1	KWT-2 [23]	97.3 ± 0.03
HTS-AT	$\mathbf{97.0 \pm 0.2}$	HTS-AT	98.0 ± 0.03

Table 2: The accuracy score results on ESC-50 dataset and Speech Command V2 (SCV2).

expect to convert tokens into activation maps for each label-class (i.e. aware of semantic meaning [16]). For strong-label datasets, we can let the model directly calculate the loss in specific time ranges. For weakly-labeled datasets, we can leverage the transformer to locate via its strong capability to capture the relation. In HTS-AT, as shown in the right of Figure 1, we modify the output structure by adding a token-semantic CNN layer after the final transformer block. It has a kernel size $(3, \frac{F}{8P})$ and a padding size $(1, 0)$ to integrate all frequency bins and map the channel size $8D$ into the event classes C. The output $(\frac{T}{8P}, C)$ is regarded as a event presence map. Finally, we average the featuremap as the final vector $(1, C)$ to compute the binary cross-entropy loss with the groundtruth labels. Apart from the localization functionality, we also expect the token-semantic module to improve the classification performance, as it considers the final output by directly grouping all tokens .

3. EXPERIMENTS

In this section, we evaluate the performance of HTS-AT in four datasets: the event classification on AudioSet [7], ESC-50 [8]; the keyword spotting on Speech Command V2 [4]; and additionally, the event detection on DESED [18].

3.1. Event Classification on AudioSet

3.1.1. Dataset and Training Detail

The AudioSet contains over two million 10-sec audio samples labeled with 527 sound event classes. In this paper, we follow the same training pipeline in [11, 12, 14] by using the full-train set (2M samples) to train our model and evaluating it on the evaluation set (22K samples). All samples are converted to mono as 1 channel by 32kHz sampling rate. We use 1024 window size, 320 hop size, and 64 mel-bins to compute STFTs and mel-spectrograms. As a result, the shape of the mel-spectrogram is $(1024, 64)$ as we pad each 1000-frame (10-sec) sample with 24 zero-frames (T=1024, F=64). The shape of the output featuremap is $(1024, 527)$ (C=527). The patch size is 4×4, the patch window length is 256 frames,

and the attention window size is 8×8. Since 8 is divisible by 64, the attention window in the first layer will not span two frames with a large time difference. The latent dimension size is D=96 and the final output latent dimension is $8D$=768, which is consistent to AST. Finally, we set 4 network groups with 2, 2, 6, 2 swin-transformer blocks respectively.

We follow [11, 12] to use the balance sampler, $\alpha = 0.5$ mix-up [19], spectrogram masking [20] with time-mask=128 frames and frequency-mask=16 bins, and weight averaging. The HTS-AT is implemented in Pytorch and trained via the AdamW optimizer (β_1=0.9, β_2=0.999, eps=1e-8, decay=0.05) with a batch size of 128 (32×4) in 4 NVIDIA Tesla V-100 GPUs. We apply a warm-up schedule by setting the learning rate as 0.05, 0.1, 0.2 in the first three epochs, then the learning rate is halved every ten epochs until it returns to 0.05. We use the mean average precision (mAP) to evaluate the classification performance.

3.1.2. Experimental Results

In Table 1, we compare our HTS-AT with different benchmark models and three self-ablated variations: (1) H: only hierarchical structure; (2) HC: with hierarchical structure and token-semantic module; and (3) HCP: (2) with pretrained vision model (the full setting). Our best setting achieves a new SOTA mAP 0.471 in a single model as a large increment from 0.459 by AST. We also ensemble six HTS-ATs with different training random seeds in the same settings to achieve the mAP as 0.487, and outperforms AST's 0.475 and 0.485. We analyze our results in two facets.

Token Semantic Module and Pretraining PSLA, AST and HTS-AT adopt the ImageNet-pretrained model, where PSLA uses the pretrained EfficientNet [25], AST uses DeiT [26], and our HTS-AT uses the swin-transformer in Swin-T/C24 setting[3] for 256×256 images. We can see that the unpretrained single HTS-AT can achieve an mAP as 0.440. It is improved to 0.453 by the addition of token semantic module, 1.8% lower than 0.471. Finally the pretrained HTS-AT achieves the new best mAP as 0.471. However, the unpretrained single AST only reflects 0.366, 9.3% lower than 0.459. These indicate that: (1) the pretrained model definitely improves the performance by building a solid prior on pattern recognition; and (2) HTS-AT shows a far better scalability to different hyperparameters than AST, since its unpretrained model can still achieve the third best performance.

---

[2]AST provides a second bigger ensemble result by using models with different patch settings, which is partially comparable with our settings.

[3]https://github.com/microsoft/Swin-Transformer

Model	Alarm	Blender	Cat	Dishes	Dog	Shaver	Frying	Water	Speech	Cleaner	Average
PANN [11]	34.3	42.4	36.3	17.6	35.8	23.8	9.3	30.6	69.7	51.0	35.1
HTS-AT	**48.6**	52.9	67.7	25.0	48.0	42.9	60.3	43.0	46.8	49.1	48.4
HTS-AT - Ensemble	47.5	55.1	**72.4**	30.9	**49.7**	41.9	**63.2**	**44.3**	51.3	50.6	50.7
Zheng et al.* [24]	41.4	54.1	**72.4**	29.4	47.8	**61.01**	49.2	33.7	**69.5**	65.5	**52.4**
Kim et al.* [24]	34.7	**59.8**	71.6	40.4	47.3	26.2	61.8	32.8	64.9	**66.7**	50.6
Lu et al.* [24]	37.1	41.4	62.5	**40.6**	39.7	46.5	46.5	34.5	54.5	46.9	45.0

Table 3: The event-based F1-scores of each class on the DESED test set. Models with ***** are from DCASE 2021 [24], which are partial references since they use extra training data and are evaluated on DESED test set and its another private subset.

Parameter Size and Training Time   When comparing the parameter size of each model, the AST has 87M parameters. And HTS-AT is more lightweight with 31M parameters, which is even compatible with CNN-based models. As for the estimated training time, PANN takes about 72 hours to converge and HST-AT takes about $20 \times 4 = 80$ hours in V-100 GPUs; and AST takes about $150 \times 4 = 600$ hours in 4 TITAN RTX GPUs[4]. The speed improvement corresponds to the less calculation and GPU memory consumption of HTS-AT, as we could feed 128 samples instead of only 12 samples in AST per batch. Therefore, we conclude that HTS-AT consumes less training time and has fewer parameters than AST's.

3.2. Evaluations on ESC-50 and Speech Command V2

3.2.1. Dataset and Training Detail

The ESC-50 dataset contains 2000 5-sec audio samples labeled with 50 environmental sound classes in 5 folds. We train the model for 5 times by selecting 4-fold (1600 samples) as training set and the left 1-fold (400 samples) as test set. And we repeat this experiment 3 times with different random seeds to get the mean performance and deviation. The Speech Command V2 contains 105,829 1-sec spoken word clips labled with 35 common word classes. It contains 84843, 9981, and 11005 clips for training, validation and evaluation. Similarly, we train our HTS-AT for 3 times to obtain the prediction results. We use the mean accuracy score (acc) for the evaluation on both datasets. For the data processing, we resample the ESC-50 samples into 32kHz and the Speech Command clips 16kHz. And we follow the same setting.

3.2.2. Experimental Results

We use our best AudioSet-pretrained HTS-AT to train on these two dataset respectively and compare it with benchmark models (also in AudioSet or extra data pretraining). Since 1-sec and 5-sec does not take the full 10-sec input trained on AudioSet, we repeat the 1-sec and 5-sec by 10 and 2 times to make it 10-sec. As shown in Table 2, the results shows that our HTS-AT achieves a new SOTA as 97.0% on ESC-50 dataset and equals the SOTA 98.0% on Speech Command V2. Our deviations are relatively smaller than AST's, indicating that HTS-AT is more stable after convergence.

3.3. Localization Performance on DESED

We additionally evaluate HTS-AT's capability to localize the sound event as start and end time in given audio samples. We use the DESED test set [18], which contains 692 10-sec test audio samples in 10 classes with the strong labels. We mainly compare our HTS-AT with PANN. We do not include AST and PSLA since AST does not directly support the event localization and the PSLA's code is not published. We also compare it partially with models in DCASE 2021 [24], nevertheless they use extra training data and are evaluated on DESED test set and its another private subset. We use the event-based F1-score on each class as the evaluation metric, implemented by a Python library `psds_eval`[5].

The F1-scores on all 10 classes in the DESED by different models are shown in Table 3. We find that HTS-AT achieves better F1-scores on 8 classes and a better average F1-score 50.7% than PANN. When compared among leaderboard models, our model still achieves some highest scores of certain classes. However, the F1-scores on Speech and Cleaner are relatively low, indicating that there are still some improvements for a better localization performance. From the above experiments, we can conclude that HTS-AT is able to produce the specific localization output via the token-semantic module, which extends the functionality of the audio transformer.

4. CONCLUSION AND FUTURE WORK

In this paper, we propose HTS-AT: a hierarchical token-semantic transformer for audio classification. It achieves a new SOTA on multiple datasets of different audio classification scenarios. Furthermore, the token-semantic module enables HTS-AT to locate the events start and end time. Experiments show that HTS-AT is a high performance, high scalability, and lightweight audio transformer. In the future, we notice that a partial strong labeled subset of AudioSet has just been released [27], we decide to conduct a detail localization training and evaluation work by HTS-AT to further explore its potential. Combining the audio classification model into more downstreaming tasks [28, 29] is also considered a future work.

---

[4]We make memories not exceed 12GB in V-100 in line with TITAN RTX.

[5]https://github.com/audioanalytic/psds_eval

5. REFERENCES

[1] Annamaria Mesaros, Toni Heittola, Tuomas Virtanen, and Mark D. Plumbley, "Sound event detection: A tutorial," *IEEE Signal Process. Mag. 2021*.

[2] Perfecto Herrera, Geoffroy Peeters, and Shlomo Dubnov, "Automatic classification of musical instrument sounds," *Journal of New Music Research*, 2010.

[3] Ke Chen, Beici Liang, Xiaoshuan Ma, and Minwei Gu, "Learning audio embeddings with user listening data for content-based music recommendation," in *ICASSP 2021*.

[4] Pete Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *CoRR*, vol. abs/1804.03209, 2018.

[5] Ke Chen, Cheng-i Wang, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Music sketchnet: Controllable music generation via factorized representations of pitch and rhythm," in *ISMIR 2020*.

[6] Hao-Wen Dong, Ke Chen, Julian J. McAuley, and Taylor Berg-Kirkpatrick, "Muspy: A toolkit for symbolic music generation," in *ISMIR 2020*.

[7] Jort F. Gemmeke, Daniel P. W. Ellis, and Dylan Freedman et al., "Audio set: An ontology and human-labeled dataset for audio events," in *ICASSP 2017*.

[8] Karol J. Piczak, "ESC: dataset for environmental sound classification," in *ACM MM 2015*. ACM.

[9] Logan Ford, Hao Tang, and François Grondin et al., "A deep residual network for large-scale acoustic scene analysis," in *Interspeech 2019*.

[10] Yun Wang, Juncheng Li, and Florian Metze, "A comparison of five multiple instance learning pooling functions for sound event detection with weak labeling," in *ICASSP 2019*.

[11] Qiuqiang Kong, Yin Cao, and Turab Iqbal et al., "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE TASLP 2020*.

[12] Yuan Gong, Yu-An Chung, and James Glass, "Psla: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE TASLP 2021*.

[13] Ashish Vaswani, Noam Shazeer, and Niki Parmar et al., "Attention is all you need," in *NeurIPS 2017*.

[14] Yuan Gong, Yu-An Chung, and James Glass, "Ast: Audio spectrogram transformer," in *Interspeech 2021*.

[15] Jia Deng, Wei Dong, and Richard Socher et al., "Imagenet: A large-scale hierarchical image database," in *CVPR 2009*.

[16] Wei Gao, Fang Wan, and Xingjia Pan et al., "Ts-cam: Token semantic coupled attention map for weakly supervised object localization," in *ICCV 2021*.

[17] Ze Liu, Yutong Lin, and Yue Cao et al., "Swin transformer: Hierarchical vision transformer using shifted windows," *CoRR*, vol. abs/2103.14030, 2021.

[18] Romain Serizel, Nicolas Turpault, Ankit Parag Shah, and Justin Salamon, "Sound event detection in synthetic domestic environments," in *ICASSP 2020*.

[19] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR 2018*.

[20] Daniel S. Park et al., "Specaugment: A simple data augmentation method for automatic speech recognition," in *Interspeech 2019*.

[21] Roman Vygon and Nikolay Mikhaylovskiy, "Learning efficient representations for keyword spotting with triplet loss," in *SPECOM 2021*. Springer.

[22] Sergey Verbitskiy and Viacheslav Vyshegorodtsev, "Eranns: Efficient residual audio neural networks for audio pattern recognition," *CoRR*, vol. abs/2106.01621, 2021.

[23] Axel Berg, Mark O'Connor, and Miguel Tairum Cruz, "Keyword transformer: A self-attention model for keyword spotting," in *Interspeech 2021*.

[24] "Dcase 2021 challenge task 4: Sound event detection and separation in domestic environments," http://dcase.community/challenge2021.

[25] Mingxing Tan and Quoc V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML 2019*. PMLR.

[26] Hugo Touvron, Matthieu Cord, and Matthijs Douze et al., "Training data-efficient image transformers & distillation through attention," in *ICML 2021*.

[27] Shawn Hershey, Daniel P. W. Ellis, and Eduardo Fonseca et al., "The benefit of temporally-strong labels in audio event classification," in *ICASSP 2021*.

[28] Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Zero-shot audio source separation through query-based learning from weakly-labeled data," in *AAAI 2022*.

[29] Ke Chen, Gus Xia, and Shlomo Dubnov, "Continuous melody generation via disentangled short-term representations and structural conditions," in *IEEE ICSC 2020*.