Audio Mamba: Bidirectional State Space Model for Audio Representation Learning

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

***Abstract*—Transformers have rapidly become the preferred choice for audio classification, surpassing methods based on CNNs. However, Audio Spectrogram Transformers (ASTs) exhibit quadratic scaling due to self-attention. The removal of this quadratic self-attention cost presents an appealing direction. Recently, state space models (SSMs), such as Mamba, have demonstrated potential in language and vision tasks in this regard. In this study, we explore whether reliance on self-attention is necessary for audio classification tasks. By introducing Audio Mamba (AuM), the first self-attention-free, purely SSM-based model for audio classification, we aim to address this question. We evaluate AuM on various audio datasets - comprising six different benchmarks - where it achieves comparable or better performance compared to well-established AST model. Code is available at: https://github.com/kaistmm/Audio-Mamba-AuM**

***Index Terms*—State Space Models, Audio Spectrogram Transformers, Audio Classification**

I. Introduction

In recent years, CNNs [1], [2] have been replaced with transformer-based architectures [3], [4], [5], [6] in a paradigm shift in deep learning, as transformers outperform convolutional neural networks. Not only does the performance of transformers exceed that of CNNs, but establishing a unified architecture among many different research fields and tasks — traditionally using completely different models — is another breakthrough [7], [8], [9], [10], [11], [12], [13], [14], [15]. Despite their success, transformers are hindered by the reliance on the computationally heavy self-attention mechanism. The $\mathcal{O}(n^2)$ cost of it is a natural concern for processing longer sequences. This limitation motivates exploration of alternative architectures, notably state-space models (SSMs) [16], [17], [18], [19], [20] which replace the self-attention mechanism with a unidirectional scanning method for time-varying input sequences to capture global context efficiently. Recently, the introduction of Mamba [16] marks a significant advancement in representation power and efficiency for both training and inference, suggesting a potential alternative to transformer-based approaches. Given the universality and scalability of transformers across various tasks, Mamba's potential, with its computational efficiency, is promising for becoming a similarly generic and versatile architecture.

* denotes equal contribution. M.H. Erol is with Computer Science Department, Stanford University. A. Senocak and J.S. Chung are with the School of Electrical Engineering, KAIST, Korea. J. Feng is with Dept. of Computer Science, UT Austin. J. S. Chung is the corresponding author (e-mail: joonson@kaist.ac.kr). This work was supported by the National Research Foundation of Korea grant funded by the Korean government (Ministry of Science and ICT, RS-2023-00212845) and the ITRC (Information Technology Research Center) support program (IITP-2024-RS2023-00259991) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

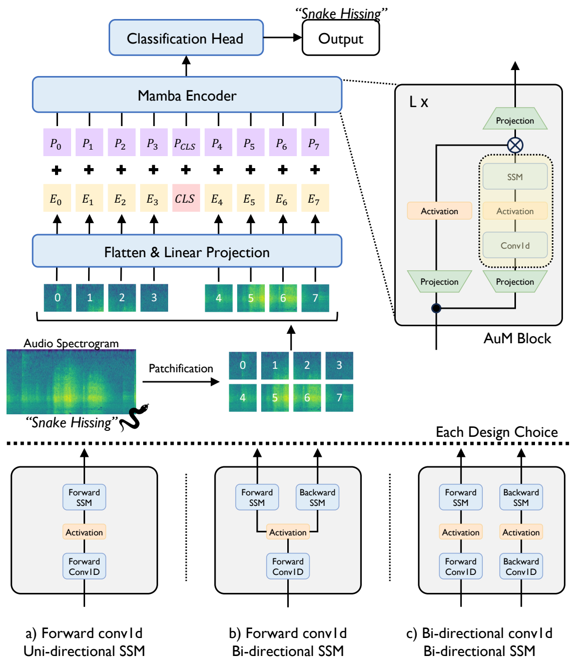
"Snake Hissing"
Classification Head
Output
Mamba Encoder
P_0 P_1 P_2 P_3 P_{CLS} P_4 P_5 P_6 P_7
E_0 E_1 E_2 E_3 CLS E_4 E_5 E_6 E_7
Flatten & Linear Projection
0 1 2 3 4 5 6 7
Audio Spectrogram
Patchification
"Snake Hissing"
L x
Projection
SSM
Activation
Conv1d
AuM Block
Each Design Choice
Forward SSM
Backward SSM
Forward Conv1D
Backward Conv1D
a) Forward conv1d Uni-directional SSM
b) Forward conv1d Bi-directional SSM
c) Bi-directional conv1d Bi-directional SSM

Fig. 1: **The proposed Audio Mamba (AuM) architecture.**

Despite the successes of SSMs in audio [20], [21], [22] and Mamba in language modeling and vision [23], [24], [25], [26], [27], the adoption of Mamba in the audio classification domain still remains unexplored. This gap motivates our work, where we introduce a novel SSM-based model, Audio Mamba - AuM, applied directly to audio spectrograms. Our approach is self-attention free, focusing purely on long sequence modeling with state space models. AuM not only achieves comparable performance to the Audio Spectrogram Transformer (AST) [6], the most prominent approach in audio classification, but also retains several advantages of transformer-based models. These include the ability to handle varying sequence lengths and the ease of transferability to other tasks. Due to the employment of state space models, reliance on self-attention is eliminated, enabling the model to operate with linear time complexity relative to sequence length and feature dimension, as opposed to AST where quadratic complexity is observed. The closest work to ours is Vision Mamba [23], which utilizes bidirectional SSM for global visual context modeling and positional embeddings for location information in a structure similar to Vision Transformers (ViT) [4]. Drawing on AST's success in applying ViT's principles to audio classification, we also draw inspiration from the findings of Vision Mamba and study the methodologies suitable for applying state space models to audio classification. To accomplish this task, we take the following steps: (1) We divide the input spectrogram into patches, which are then projected into patch embedding tokens. (2) We add an additional learnable classification token to the sequence of patch tokens, specifically in the middle. (3) The Audio Mamba Encoder blocks process these token sequences in both forward and backward directions with SSM modules. (4) The classification token is utilized to train the model on the supervised audio classification task and also for making predictions in the inference stage. We summarize the contributions of our work as follows:

- We introduce Audio Mamba (AuM) for processing audio spectrograms, which follows a similar generic structure to Audio Spectrogram Transformer (AST). However, AuM uniquely utilizes bidirectional state space models (SSM) to handle tokens in both forward and backward directions.
- By eliminating self-attention modules, AuM achieves linearly scaled resource consumption when evaluated with long audio sequences.
- Our comprehensive experiments across six diverse datasets — AudioSet [28], AudioSet Balanced, VGGSound [29], VoxCeleb [30], Speech Commands V2 [31], and Epic-Sounds [32] — show that AuM delivers performance that is comparable to or exceeds the most prominent audio classification method AST.

II. Audio Mamba

A. Flow of the Architecture

The Audio Mamba (AuM), as depicted in Fig. 1, begins by transforming an input audio waveform into an audio spectrogram $X \in \mathbb{R}^{F \times T}$, where F and T represent the frequency and time dimensions, respectively. The spectrogram is partitioned into a sequence of M' square patches $S \in \mathbb{R}^{M' \times p \times p}$, with p denoting the side length of each patch and M' calculated as $M' = (F/p) \times (T/p)$. Each individual patch S_i is subsequently flattened into a one-dimensional vector $S_i \in \mathbb{R}^{p^2}$, and through a linear projection, it is embedded into a D-dimensional space, yielding $E_i \in \mathbb{R}^D$. This process is done by the patch embedding layer. Afterward, a special learnable classification token, denoted as $CLS \in \mathbb{R}^D$, is inserted into the middle of the sequence, leading to an augmented embedding sequence $E \in \mathbb{R}^{M \times D}$ where $M = M' + 1$. To encode the position of each element within the sequence, learnable positional embeddings $P \in \mathbb{R}^{M \times D}$ are added, resulting in the token sequence $T \in \mathbb{R}^{M \times D}$. This token sequence is then processed by the Audio Mamba encoder which consists of L stacked blocks, each of which retains the dimensionality of its input. Thus, the encoder transforms T into an output sequence $T' \in \mathbb{R}^{M \times D}$. The modified representation of the classification token $T'_{M'/2}$ is then conveyed to the classification head. We note the adoption of standard techniques like patch partitioning, positional embeddings, and the CLS token, commonly used in models like ViT [4], AST [6], and ViM [16], to maintain structural similarity with AST. These choices are validated by our empirical results in Table V in supp. material.

B. Architecture Details

Aiming to establish itself as a generic architecture, AuM shares similarities with AST [6]. However, AuM distinguishes itself with unique architectural and operational characteristics, notably being a SSM-based self-attention-free model.

Preliminaries. State space models (SSMs) are linear time-invariant systems that aim to model a continuous system which maps a time dependent input sequence $x(t) \in \mathbb{R}$ to an output $y(t) \in \mathbb{R}$ through maintaining a hidden state $h(t) \in \mathbb{R}^N$. Such a system could be represented with the following equation:

$$\begin{aligned} h'(t) &= Ah(t) + Bx(t), \\ y(t) &= Ch(t) \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{N \times N}$, $B \in \mathbb{R}^{N \times 1}$ and $C \in \mathbb{R}^{1 \times N}$ are time-invariant parameters. With the primary goal of adapting the model to deep learning algorithms, a discretization process is applied, which transforms the continuous parameters A and B through a discretization rule into $\bar{A}$ and $\bar{B}$, respectively. These discretized parameters are then substituted for A and B, leading to the following discretized formulation of the system:

$$\begin{aligned} h_t &= \bar{A}h_{t-1} + \bar{B}x_t, \\ y_t &= Ch_t. \end{aligned} \tag{2}$$

This linear time-invariant system can be extended to work for inputs with larger dimensions (such as $x_t \in \mathbb{R}^D$) by treating each dimension as a separate SSM stream. Also, it can be processed efficiently either as a linear recurrence or through a global convolution [16]. However, its time-invariant nature limits the modeling capabilities for certain data types. Mamba enhances these models by converting time-invariant parameters into time-variant ones, through an efficient parameter derivation from the time-varying inputs. For instance, in the Forward SSM module of a Mamba block (Fig. 1 (a)), the input sequence $x \in \mathbb{R}^{M \times D}$ after the application of Conv1D is utilized to convert each time-invariant parameter A, B, and C into specific corresponding parameters A'_t, B'_t, and C'_t for each element x_t. These parameters help the Forward SSM module process the sequence from the beginning to the end in a unidirectional manner by selectively updating its hidden state to capture relevant information from the input sequence. More details are available in [16].

Bidirectional Mamba Encoder. Mamba's unidirectional scan is useful for modeling causal sequential data, but learning from 2D data benefits from multi-directional processing [23], [24], [25]. For instance, in learning from visual data, the ViM architecture [23] modifies the original Mamba block in Fig. 1 (a) to Fig. 1 (c) to include an additional direction for feature extraction (Backward Conv1D) and scanning of the input sequence (Backward SSM). The sum of the token transformations from each direction enables multi-directional and spatial-aware processing. Similarly, AuM adopts the design strategy shown in Fig. 1 (b) by adding an extra backward scanning direction to the original Mamba block. This approach utilizes the same convolved features while adapting both the forward and backward SSM parameters into their time-variant (input-dependent) versions for scanning. Likewise ViM, AuM also sums these token transformations to model the global context in a spatial aware manner, mirroring the functionality of self-attention mechanism in transformers for modeling global context. More details are in Algorithm 1 in supp. material.

TABLE I: **Results of from-scratch training of AST and AuM base models across various datasets.**

Model	**AudioSet (mAP)**	**AS-20K (mAP)**	**VGGSound (Acc.)**	**VoxCeleb (Acc.)**	**Speech Comm. V2 (Acc.)**	**Epic-Sounds (Acc.)**
AST-B/16	29.10± 0.07	10.41 ± 0.32	37.25 ± 0.31	22.44 ± 0.19	85.27 ± 1.07	**44.76** ± 0.20
AuM-B/16	**32.43** ± 0.31 (+3.33)	**13.28** ± 1.07 (+2.87)	**42.58** ± 0.28 (+5.33)	**28.34** ± 3.38 (+5.90)	**91.58** ± 3.17 (+6.32)	44.17 ± 0.58 (-0.60)

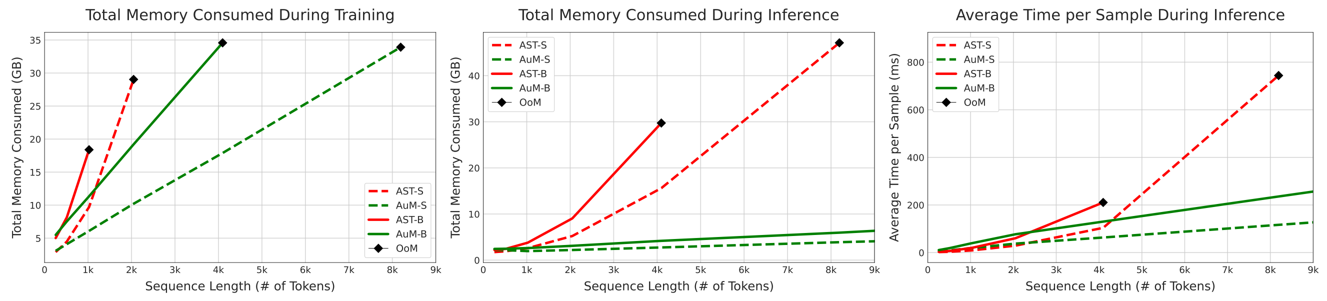

Fig. 2: **Empirical evaluation of memory and time consumption for AST and AuM small/base models.**

Classification Token. Unlike transformers in their pure form, which are permutation invariant when processing the input sequence, AuM block is sensitive to the order of the input sequence because both feature extraction (Conv1D) and SSMs are input-order-sensitive operations. Consequently, in addition to scanning directions, the placement of the classification token within the input sequence becomes critical for the learning. Similarly to ViM, after the patch embedding layer, AuM strategically inserts the classification token at the midpoint of the input sequence. This setup has shown improved performance in bidirectional processing setup, as demonstrated in ViM, and the ablation study conducted in Section III.

III. Experiments

A. Datasets and Evaluation Metrics

Datasets. Our experiments utilize: Audioset Full / Balanced [28], VGGSound [29], VoxCeleb [30], Speech Commands-V2 [31], and EPIC-SOUNDS [32] datasets.

Evaluation metrics. We utilize mean average precision (mAP) for AudioSet experiments due to the existence of multiple labels per sample. For the remaining datasets, top-1 classification accuracy (Acc) is used (samples have a single label).

B. Comparison to AST on Standard Benchmarks

In this section, we conduct a comparative analysis of AuM against the AST model. Both models use base backbones, AuM-B/16 and AST-B/16. To ensure a fair comparison, we follow the same training and experimental settings as AST, including maintaining fixed audio lengths for each dataset, detailed in the supp. material. Importantly in this experiment, neither AuM nor AST utilized pretraining weights from other models (AST is initialized with weights from the Vision Transformer (ViT) model pretrained on ImageNet in the original paper [6]) to ensure a pure comparison of these two different architectures. We repeat each experiment three times with the same setup but different random seeds and report the results with the mean and standard deviation in Table I. Our proposed *AuM generally achieves better performance* in this experimental setup. This indicates that AuM, in its pure form, can serve as a potential alternative to the AST by offering better computational efficiency and not using self-attention.

TABLE II: **Results of ablation on the design choices.** Architectural choices (under the settings column) refer to the block types in Fig. 1, the location of the classification token is indicated through the columns: Head, Mid and End per dataset.

Setting	**AS-20K (mAP)**			**VGGSound (Acc.)**		
	Head	**Mid**	**End**	**Head**	**Mid**	**End**
Fo-Fo (a)	0.48± 0.00	11.73± 0.47	11.90± 1.05	0.33± 0.00	35.05± 0.80	39.09± 0.56
Fo-Bi (b)	13.57± 0.09	**13.81**± 0.32	12.35± 0.33	40.91± 0.47	**42.58**± 0.28	40.41± 0.15
Bi-Bi (c)	4.97± 0.13	9.69± 0.43	11.11± 0.66	34.22± 0.26	36.48± 0.46	41.09± 0.16

C. Comparison to AST on Efficiency

Transformer-based audio classification models are computationally demanding (quadratic complexity), particularly with lengthy audio and high-dimensional data. SSM-based models stand out for their computational and memory efficiency. We compare AuM to AST from an efficiency perspective using a single A6000 GPU. We feed audios with corresponding lengths for every given token number to the models to simulate the speed and GPU memory comparison, visualized in Figure 2 where AuM demonstrates clear computation and memory efficiency. For example, the AuM-Base model that uses 20 seconds of audio (1024 tokens) for training consumes as little GPU memory as the AST-Small model. Additionally, while AuM-B can be trained with 80-second audios, AST-B runs out of memory with only 20-second audios. Moreover, AuM is 1.6 times faster in the inference stage than AST at 4096 number of tokens, with a growing rate as the token count increases. All these results indicate that AuM exhibits a trend of linear scaling with respect to sequence length.

D. Ablation Study on Design Choices

We conduct experiments to verify our design choices and do further analysis. We study the following strategies regarding the direction of SSM and conv1Ds modules:

<u>**AuM-Fo**rwardConv1D-**Fo**rwardSSM:</u> This choice, which is the default Mamba block, directly applies the AuM Block with only a forward SSM (refer to Figure 1 (a)).

TABLE III: **Results of ImageNet pretrained initializations of AST and AuM small models across various datasets.** Note that the setup of AuM-S is (c) in Fig. 1 due to the unavailability of the ViM weights for our preferred setup (b).

Model	**AudioSet (mAP)**	**AS-20K (mAP)**	**VGGSound (Acc.)**	**VoxCeleb (Acc.)**	**Speech Comm. V2 (Acc.)**	**Epic-Sounds (Acc.)**
AST-S	**40.32** ± 0.08	**29.20** ± 0.11	**49.51** ± 0.06	39.70 ± 1.83	97.38 ± 0.07	52.42 ± 0.11
AuM-S (c)	39.68 ± 0.06 (-0.64)	28.89 ± 0.20 (-0.31)	49.43 ± 0.18 (-0.07)	**40.58** ± 1.11 (+0.89)	**97.51** ± 0.08 (+0.13)	**52.90** ± 0.40 (+0.48)

TABLE IV: **Results of Audioset pretrained initializations of AST and AuM base models across various datasets.**

Model	**VGGSound (Acc.)**	**VoxCeleb (Acc.)**	**Speech Comm. V2 (Acc.)**	**Epic-Sounds (Acc.)**
AST-B/16	44.17 ± 0.14	**46.25** ± 1.08	90.37 ± 0.06	46.62 ± 0.04
AuM-B/16	**46.61** ± 0.18 (+2.44)	40.72 ± 1.11 (-5.53)	**94.78** ± 0.04 (+4.41)	**48.18** ± 0.13 (+1.57)

AuM-ForwardConv1D-**Bi**DirectionalSSM: This is the design of our final model, which applies an additional backward SSM to the previous design choice (refer to Figure 1 (b)).

AuM-Bi DirectionalConv1D-**Bi**DirectionalSSM: This variant adds another Conv1D in the backward direction to feed the output of this module to the backward SSM, making each SSM module a separate stream. Vision Mamba (ViM) adopts a similar design as the default choice (see Figure 1 (c)).

Moreover, the position of the class token is ablated for each variant above. To save computational time, we primarily conduct ablation studies by training our model on AudioSet Balanced (AS-20K) and VGGSound, presented in Table II.

Impact of bidirectional SSM. To understand the impact of SSM directions, we analyze the performance of model variants with different directional SSM modules. The results show that bidirectional variants (forward and backward SSM modules together) overall perform better than the forward-only variant.

Impact of direction of conv1D. We conduct a controlled experiment between two bidirectional methods: **AuM-Fo-Bi** and **AuM-Bi-Bi**, where the only difference is the presence of an additional backward Conv1D. As shown in Table II, our design choice, which omits the backward Conv1D, generally performs better. We hypothesize that processing a single input sequence (output from only the forward Conv1D) is more effective and natural for scanning in both forward and backward directions to understand entire context, compared to providing separate inputs to each directional SSM module and scanning in only one direction accordingly.

Impact of the class token position. Our extensive experiments reveal that positioning the class token in the middle of the sequence is the most suitable choice for our design. The position of the class token is a crucial decision, as each variant exhibits a different optimal location for its use, which greatly impacts performance. Notably, the forward-only SSM collapses when the class token is placed at the beginning of the sequence (head class token). This is expected, since subsequent sequence information is not incorporated into the class token.

E. Impact of Pre-Training

Out-of-domain pre-training. Initializing audio models with ImageNet pre-trained weights has become popular for audio classification [6], [33]. Specifically, AST demonstrates a significant performance improvement when initialized with supervised ImageNet weights (obtained from ViT and its variants) compared to training from scratch. As presented in Table I, our main results exclude models with pretraining to provide a clear comparison between these two architectures. One might question why such results are not displayed. To the best of our knowledge, no released ViM Base model weights comparable to ViT weights for the AST model are available, preventing us from conducting this experiment. However, we still analyze AuM and AST with out-of-domain pre-training weights by using the only available small-sized Vision Mamba model, ViM-S, and compare the AuM-S and AST-S models. Notably, ViM-S is in the **Bi-Bi** variant, but our ideal choice for AuM, as highlighted in Sec. II and III-D, is **AuM-Fo-Bi**. Due to unavailability, we use the **AuM-Bi-Bi** (c) variant of our model. The findings in Table III reveal that both AuM-S (c) and AST-S models perform similarly. We believe that with the proper variant (b instead of c), our model could outperform AST, as it does without vision domain pre-trained weights.

From-scratch audio-only pre-training. After analyzing ImageNet initialization, we also explore using AudioSet-trained weights of base models (from Table I) as in-domain pre-training to initialize both AuM-B and AST-B. Here, unlike in the previous section, our model uses weights from a model that is architecturally identical to ours. The results, shown in Table IV, indicate that in-domain pre-training benefits both models, enhancing their performance. In this setting, AuM outperforms AST, except on the VoxCeleb dataset.

IV. Conclusion

In this work, we introduce Audio Mamba (AuM), the first architecture for audio classification that is free from self-attention and purely based on state space models. Our extensive experiments highlight AuM's efficiency in terms of computational and memory use, as well as its competitive performance against the well-established Audio Spectrogram Transformers (AST). Considering its similarity to AST structure regarding patchifying the input spectrogram, adding positional embeddings, and processing the information sequentially, it shows great potential to become an alternative generic audio backbone. By eliminating costly self-attention and efficiently processing long sequences, we believe AuM brings an important contribution to the audio field for future potential applications. The ability to handle lengthy audio is increasingly crucial, especially with the rise of self-supervised multimodal learning [13], [14], [15] and generation that leverages in-the-wild data and Automatic Speech Recognition. Furthermore, AuM could be employed in self-supervised learning setups like Audio Masked Auto Encoders [34], [35] or multimodal learning tasks such as Audio-Visual pretraining [7], [8], [13] or Contrastive Language-Audio Pretraining [9], [10], [11].

References

[1] 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," *TASLP*, 2020.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. ICLR*, 2021.

[5] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proc. ICML*, 2021.

[6] Y. Gong, Y.-A. Chung, and J. Glass, "AST: audio spectrogram transformer," in *Proc. Interspeech*, 2021.

[7] Y. Gong, A. H. Liu, A. Rouditchenko, and J. Glass, "Uavm: Towards unifying audio and visual models," *IEEE Signal Processing Letters*, 2022.

[8] Y. Gong, A. Rouditchenko, A. H. Liu, D. Harwath, L. Karlinsky, H. Kuehne, and J. Glass, "Contrastive audio-visual masked autoencoder," in *Proc. ICLR*, 2022.

[9] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *Proc. ICASSP*, 2023.

[10] Y.-J. Shih, H.-F. Wang, H.-J. Chang, L. Berry, H.-y. Lee, and D. Harwath, "Speechclip: Integrating speech with pre-trained vision and language model," in *IEEE Spoken Language Technology Workshop (SLT)*, 2023.

[11] H.-H. Wu, P. Seetharaman, K. Kumar, and J. P. Bello, "Wav2clip: Learning robust audio representations from clip," in *Proc. ICASSP*, 2022.

[12] A. Nagrani, S. Yang, A. Arnab, A. Jansen, C. Schmid, and C. Sun, "Attention bottlenecks for multimodal fusion," in *NeurIPS*, 2021.

[13] P. Morgado, Y. Li, and N. Nvasconcelos, "Learning representations from audio-visual spatial alignment," in *NeurIPS*, 2020.

[14] J.-B. Alayrac, A. Recasens, R. Schneider, R. Arandjelović, J. Ramapuram, J. De Fauw, L. Smaira, S. Dieleman, and A. Zisserman, "Self-supervised multimodal versatile networks," in *NeurIPS*, 2020.

[15] H. Akbari, L. Yuan, R. Qian, W.-H. Chuang, S.-F. Chang, Y. Cui, and B. Gong, "Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text," in *NeurIPS*, 2021.

[16] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[17] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *Proc. ICLR*, 2022.

[18] J. T. Smith, A. Warrington, and S. W. Linderman, "Simplified state space layers for sequence modeling," in *Proc. ICLR*, 2023.

[19] D. Y. Fu, T. Dao, K. K. Saab, A. W. Thomas, A. Rudra, and C. Re, "Hungry hungry hippos: Towards language modeling with state space models," in *Proc. ICLR*, 2023.

[20] C. Chen, C.-H. H. Yang, K. Li, Y. Hu, P.-J. Ku, and E. S. Chng, "A neural state-space model approach to efficient speech separation," in *Proc. Interspeech*, 2023.

[21] K. Goel, A. Gu, C. Donahue, and C. Ré, "It's raw! audio generation with state-space models," in *International Conference on Machine Learning*. PMLR, 2022.

[22] G. Zhu, Y. Yan, J.-P. Caceres, and Z. Duan, "Transcription free filler word detection with neural semi-crfs," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[23] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, "Vision mamba: Efficient visual representation learning with bidirectional state space model," in *Proc. ICML*, 2024.

[24] Y. Yang, Z. Xing, and L. Zhu, "Vivim: a video vision mamba for medical video object segmentation," *arXiv preprint arXiv:2401.14168*, 2024.

[25] Y. Liu, Y. Tian, Y. Zhao, H. Yu, L. Xie, Y. Wang, Q. Ye, and Y. Liu, "Vmamba: Visual state space model," *arXiv preprint arXiv:2401.10166*, 2024.

[26] J. Ma, F. Li, and B. Wang, "U-mamba: Enhancing long-range dependency for biomedical image segmentation," *arXiv preprint arXiv:2401.04722*, 2024.

[27] Z. Xing, T. Ye, Y. Yang, G. Liu, and L. Zhu, "Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation," *arXiv preprint arXiv:2401.13560*, 2024.

[28] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "AudioSet: An ontology and human-labeled dataset for audio events," in *Proc. ICASSP*, 2017.

[29] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "VGGSound: A large-scale audio-visual dataset," in *Proc. ICASSP*, 2020.

[30] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "VoxCeleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, 2020.

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

[32] J. Huh, J. Chalk, E. Kazakos, D. Damen, and A. Zisserman, "EPIC-SOUNDS: A Large-Scale Dataset of Actions that Sound," in *Proc. ICASSP*, 2023.

[33] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "HTS-AT: a hierarchical token-semantic audio transformer for sound classification and detection," in *Proc. ICASSP*, 2022.

[34] P.-Y. Huang, H. Xu, J. Li, A. Baevski, M. Auli, W. Galuba, F. Metze, and C. Feichtenhofer, "Masked autoencoders that listen," in *NeurIPS*, 2022.

[35] A. Baade, P. Peng, and D. Harwath, "MAE-AST: masked autoencoding audio spectrogram transformer," in *Proc. Interspeech*, 2022.

[36] D. Damen, H. Doughty, G. M. Farinella, , A. Furnari, J. Ma, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100," *IJCV*, 2022.

Supplementary Material

Algorithm 1 Processing of the AuM Fo-Bi Block

```
1:  Input: Token sequence $T_l$ : (B, M, D)
2:  Output: Transformed token sequence $T_{l+1}$ : (B, M, D)
3:  Define:
4:      $A$ : (E, N)
5:      $A_b$ : (E, N)
6:      $\Delta_{\text{bias}}$ : (E)
7:  Do:
8:      // layer normalization
9:      $T'_l$ : (B, M, D) $\leftarrow \text{Norm}(T_l)$
10:     // expand the embedding dimension
11:     $x$ : (B, M, E) $\leftarrow \text{Linear}_x(T'_l)$
12:     $z$ : (B, M, E) $\leftarrow \text{Linear}_z(T'_l)$
13:     // 1D convolution along the sequence
14:     $x'$ : (B, M, E) $\leftarrow \text{SiLU}(\text{Conv1D}(x))$
15:     // deriving parameters from the input
16:     $B$ : (B, M, N) $\leftarrow \text{Linear}_B(x')$
17:     $C$ : (B, M, N) $\leftarrow \text{Linear}_C(x')$
18:     // for discretization
19:     $\Delta$ : (B, M, E) $\leftarrow \log(1 + \exp(\text{Linear}_\Delta(x') + \Delta_{\text{bias}}))$
20:     // discretize the SSM params
21:     $\bar{B}$ : (B, M, E, N) $\leftarrow \Delta \otimes B$
22:     $\bar{A}$ : (B, M, E, N) $\leftarrow \Delta \otimes A$
23:     $\bar{A}_b$ : (B, M, E, N) $\leftarrow \Delta \otimes A_b$
24:     // forward SSM
25:     $y$ : (B, M, E) $\leftarrow \text{SSM}(x', \bar{A}, \bar{B}, C) \odot \text{SiLU}(z)$
26:     // reverse the param. sequences for the backward SSM
27:     $x'_b,\ z_b$ : (B, M, E) $\leftarrow \text{flip}(x'),\ \text{flip}(z)$
28:     $\bar{B}_b,\ C_b$ : (B, M, E, N) $\leftarrow \text{flip}(\bar{B}),\ \text{flip}(C)$
29:     // backward SSM
30:     $y_b$ : (B, M, E) $\leftarrow \text{SSM}(x'_b, \bar{A}_b, \bar{B}_b, C_b) \odot \text{SiLU}(z_b)$
31:     // combine the forward / backward outputs
32:     $y'$ : (B, M, E) $\leftarrow y + \text{flip}(y_b)$
33:     // back to normal embedding dim. and add residual
34:     $T_{l+1}$ : (B, M, D) $\leftarrow \text{Linear}_T(y') + T_l$
35: return $T_{l+1}$ : (B, M, D)
```

A. Dataset Details

AudioSet [28] is a dataset with a wide array of audio samples, each marked with a set of labels. It includes over 2 million 10 seconds long audio clips with a total of 527 labels. The balanced set on the other hand is curated from the full set, consisting of 20K samples. **VGGSound** [29] contains ~200k videos of 10 seconds each, annotated with 309 diverse sound categories. **VoxCeleb** [30] is a dataset focused on audio-visual representations of human speech, featuring 1,251 speakers and around 145k speech instances. **Speech Commands-V2** [31] comprises ~105k audio recordings, each with a duration of 1 second, and includes 35 widely recognized speech commands. Finally, **EPIC-SOUNDS** [32], part of EPIC-KITCHENS-100 [36], comprises 75.9k audio segments from egocentric videos, labeled across 44 classes, focusing on actions discernible by sound, such as object collisions with material annotations.

TABLE V: Performance comparison of AuM-B model on AS-20K under different architectural choices.

Input Representation	**Positional Embedding**	**Patch Size**	**Result (mAP)**
CLS Token	✓	16x16	13.81 ± 0.32
Mean Pool	✓	16x16	11.55 ± 0.27 (-2.26)
CLS Token	✗	16x16	13.87 ± 0.30 (+0.06)
CLS Token	✓	128x2	11.95 ± 0.22 (-1.86)

B. Training Setup

We train all the models (AuM and AST) of all sizes (Base and Small) across six different datasets by following the training setup shown in Table VI. This training setup and methodology follows that of AST to ensure a fair comparison in assessing the effectiveness of AuM as viable alternative.

C. Model Details

The configurations and details of all the models used throughout the experiments are shown in Table VII. Note that AuM models employ the same hyperparameters as their corresponding ViM counterparts [23]. For fairness, the ImageNet pretrained weights are sourced from DeiT and ViM models with similar architectural properties and training setups. Additionally, Algorithm 1 provides the pseudocode for how the preferred AuM block (Fo-Bi) processes and transforms an input token sequence from a preceding layer. As indicated in Table VII, the dimensionality variables B, M, D, E, and N represent the batch size, token sequence length, embedding dimension, expanded embedding dimension (during processing within the AuM block), and the hidden state size, respectively. Furthermore, it can be observed that, following the single thread of a Conv1D operation, all input-dependent parameters except the hidden state transformation parameter A are shared during both the forward and backward SSM modules, highlighting the unique characteristics of this block type.

D. Empirical Analysis for Architectural Choices

Transformer-based approaches adopt well-known standard techniques such as square patchification, positional embeddings, and classification token (CLS token). To maintain architectural similarity with the AST, we use the same standard techniques in Audio Mamba (AuM). The empirical analysis presented in Table V of this section further validates that by adopting the same standard techniques as AST, AuM can maintain architectural consistency while simply replacing self-attention with the SSM-based model. First, we compared the use of CLS tokens with Mean Pooling, where instead of learning a specific token for classification, the average of all tokens is used to classify the input. As expected, the results indicate that the CLS token is an important component for optimal performance. Secondly, we ablated the use of positional embeddings, which showed a negligible difference, suggesting that positional embeddings can be retained without significant impact. Lastly, we evaluated the commonly used square patch partitioning to rectangular frame patches, confirming that square patchification remains the optimal choice.

TABLE VI: Training setup comparison across different datasets. Here, '*' indicates that we follow the official learning rate scheduler presented in the Epic Sounds paper.

Setting	Audioset	AS-20K	VGGSound	VoxCeleb	Speech Comm. V2	Epic Sounds
Optimizer	Adam(wd=5e-7,betas=(0.95, 0.999))					
Patch Size / Stride	16 x 16 / (16, 16)					
Weighted Average	No					
Ensembling	No					
Multilabel	Yes		No			
Balanced Sampling	Yes	No				
Spec. Window Size (ms)	25					10
Spec. Hop Size (ms)	10					5
Warm-up Duration	1000 steps					2 Epochs
Max Audio Length (s)	10				1	10
Audio Sampling Rate	48k		16k			24k
Batch Size	12				128	12
Spectrogram Size	128x1024				128x128	128x1024
SpecAug (freq. / time)	48 / 192				48 / 48	48 / 192
Loss Function	BCE			CE	BCE	CE
Mixup	0.5		0		0.6	0
Epochs	5	25	20		30	
LR Sched. Type	MultiStepLR(start / step / decay)					LambdaLR*
LR Sched. Params	2 / 1 / 0.5	10 / 5 / 0.5	5 / 2 / 0.75		5 / 1 / 0.85	
Dataset Mean for Norm.	-4.268		-5.077	-3.761	-6.846	No
Dataset Std. for Norm.	4.569		4.453	4.201	5.565	
Base LR	1e-5	5e-5	1e-5		2.5e-4	1e-5

TABLE VII: Model details including parameter counts, pretrained weight sources, and various dimensions. The models are designed for training on AudioSet.

Model	**# Params**	**ImageNet Weight Source**	**Depth (L)**	**Embed Dim (D)**	**Expanded Embed Dim (E)**	**SSM State Dimension (N)**
AST-S	22.18M	DeiT-Small-patch16-224[1]	12	384	✗	✗
AST-B	86.82M	✗	12	768	✗	✗
AuM-S (Fo-Bi)	23.94M	✗	24	384	2 * 384	16
AuM-S (Bi-Bi)	25.5M	ViM-small-imgnet 80.5[2]	24	384	2 * 384	16
AuM-B	92.1M	✗	24	768	2 * 768	16

[1] https://huggingface.co/facebook/deit-small-patch16-224
[2] https://huggingface.co/hustvl/Vim-small-midclstok