

An overview of neural architectures for self-supervised audio representation learning from masked spectrograms

Sarthak Yadav^{*†}, Sergios Theodoridis^{*‡} *Life Fellow, IEEE* and Zheng-Hua Tan^{*†} *Senior Member, IEEE*

Abstract—In recent years, *self-supervised* learning has amassed significant interest for training deep neural representations without labeled data. One such self-supervised learning approach is *masked spectrogram modeling*, where the objective is to learn semantically rich contextual representations by predicting removed or hidden portions of the input audio spectrogram. With the *Transformer* neural architecture at its core, masked spectrogram modeling has emerged as the prominent approach for learning general purpose audio representations, a.k.a. audio foundation models. Meanwhile, addressing the issues of the *Transformer* architecture, in particular the underlying *Scaled Dot-product Attention* operation, which scales quadratically with input sequence length, has led to renewed interest in recurrent sequence modeling approaches. Among them, *Selective structured state space models* (such as Mamba) and *extended Long Short-Term Memory* (xLSTM) are the two most promising approaches which have experienced widespread adoption. While the body of work on these two topics continues to grow, there is currently a lack of an adequate overview encompassing the intersection of these topics. In this paper, we present a comprehensive overview of the aforementioned research domains, covering masked spectrogram modeling and the previously mentioned neural sequence modeling architectures, Mamba and xLSTM. Further, we compare Transformers, Mamba and xLSTM based masked spectrogram models in a unified, reproducible framework on ten diverse downstream audio classification tasks, which will help interested readers to make informed decisions regarding suitability of the evaluated approaches to adjacent applications.

Index Terms—deep learning, audio representation learning, self-supervised learning, masked spectrogram modeling, audio foundation models, Transformers, state space models, xLSTM.

I. INTRODUCTION

IN recent years, deep representation learning has observed an immense increase in scale, both in terms of parametric complexity of deep neural networks (DNNs) as well as the amount of data required to train them. As such, going beyond the realm of supervised learning on labeled data and leveraging the vast amounts of unlabeled data available at our disposal has garnered significant interest. While several such methods exist, self-supervised learning (SSL) has emerged as a prominent approach for training deep neural representations without labeled data. Lying somewhere in the middle of the supervised learning-unsupervised learning spectrum, the training objective in SSL is to solve a *pretext* supervised learning task utilizing labels automatically generated from a subset of the data itself. The sole purpose of these designed pretext tasks is to guide the learning of contextually and semantically rich representations that are useful for subsequent practical *downstream* applications. While several methods for learning

self-supervised representations have been proposed, predicting removed/hidden portions of input data, a.k.a. *masked predictive modeling*, which was originally proposed for natural language processing (NLP) [1], is one of the most popular SSL approaches. Together with the *Transformer* architecture [2], masked predictive models have enabled several key breakthroughs across multiple application domains, such as NLP [1], [3]–[5], computer vision [6]–[9], as well as speech [10]–[15] and audio [16]–[19] representations.

Although a very large proportion of recent works focus on learning self-supervised speech representations, learning general audio representations, viz. audio foundational models, has recently garnered a lot of interest. As opposed to the predominantly local information modelled by speech representations, audio foundation models need to capture a mix of local and global dependencies in the data. Audio foundation models are often pretrained on large audio datasets to learn a general representation that can capture a wide variety of acoustic and semantic information and can be adapted to a specific downstream task without significant training efforts. These downstream tasks can span linguistic, paralinguistic, music/pitch perception and general audio classification domains. Masked predictive modeling on audio spectrogram patches, which we hereafter refer to as masked spectrogram modeling, has emerged as a prominent approach for learning general audio representations. In masked spectrogram modeling, input audio spectrograms are divided into (generally) non-overlapping patches. Similar to MLMs, masked spectrogram models (MSMs) comprise of a *Transformer* encoder that captures contextual representations from the partially masked input patches, and a decoder that reconstructs masked portions of the input based on the encoded representations. After pretraining, the decoder is usually discarded. Based on how input patches are masked and presented to the encoder, MSMs can be broadly categorized into two types: (i) the SSAST [16], [20] class of models, where patches to be hidden are replaced with a learnable mask token before being fed to the encoder; and (ii) the Masked Autoencoder (MAE) [7], [17], [21], [22] class of models, where patches to be hidden are *removed* from the input sequence fed to the encoder. Both of these architectures have distinct characteristics and trade-offs. SSAST based models are simpler and closer to the original MLMs, and since the encoder is masking-aware, a computationally cheaper decoder can be employed. On the other hand, the MAE design allows for asymmetrically large encoders to be paired with much smaller decoders, as the larger computation load for the encoder is offset by smaller input sequence lengths when pretraining, making them highly scalable. However, while the masking methodology forces MAE encoders to learn better

^{*}Department of Electronic Systems, Aalborg University, Aalborg, Denmark

[†]Pioneer Centre for Artificial Intelligence, Denmark

[‡]National and Kapodistrian University of Athens, Athens, Greece

contextual representations, the MAE encoder design is closely coupled with the Transformer architecture and the underlying SDPA operation that treats input as elements of a set, which makes it possible to drop elements in the input sequence. This masking and encoding methodology is incompatible with traditional sequence modeling paradigms.

While Transformers have emerged as the eminent neural architecture and are the backbone of all of the above-mentioned MSMs, they are not without drawbacks. Scaled dot-product attention, the operation at the heart of the Transformer, has quadratic complexity with respect to input sequence length, and thus scales very poorly to large sequences. Further, standard scaled dot-product attention also necessitates storing the entire key-value (KV) cache for retrieval operations, thus imposing high memory requirements. Significant research efforts have been made to address these shortcomings, such as finding linear approximations for self-attention [23], [24], optimized hardware-aware implementations [25], [26] and bypassing attention entirely, either through new token mixing techniques [27], [28], or by revisiting the classical recurrent neural network paradigm. Recently, two such approaches have garnered significant research interest: state space models (SSMs) [29], and the extended long-short term memory (xLSTM) neural architecture [30]. SSMs are a class of sequence models at the intersection of convolutional neural networks, recurrent neural networks and classical state spaces from control theory, and are governed by a set of first-order differential equations. While several SSM variants [31]–[35] have been proposed, the most prominent is selective structured state spaces, a.k.a. Mamba [36], [37], which has demonstrated competitive performance against Transformer based alternatives in several domains, and have also seen widespread application in the audio domain [20], [38]–[42]. The xLSTM architecture, as the name suggests, is an extension of the LSTM [43] sequence modeling approach. xLSTMs address the critical weak points in LSTMs by integrating recent technological advances resulting from years of Transformer and large language modeling research, and have demonstrated superior sequence length extrapolation capabilities compared to Transformers. xLSTMs have seen widespread adoption in several domains, including audio and speech [44], [45]. Both Mamba and xLSTMs have been used for training audio foundation models in a masked spectrogram modeling paradigm.

Despite the large interest in masked spectrogram modeling and neural sequence modeling architectures beyond Transformers, a comprehensive overview article that investigates MSMs while incorporating a study of Transformers, Mamba and xLSTMs is still lacking. This paper presents a systematic overview of the aforementioned research domains. The objective of this paper is to enable the reader to navigate the scientific landscape as well as to get a good grasp of the fundamental concepts at the intersection of these topics, and hopefully inspire new lines of research in the field. We provide a comprehensive, systematic description of masked spectrogram modeling, both SSAST and the MAE class of MSMs and the various underlying components. We also provide a comprehensive coverage and review of the Mamba and the xLSTM neural sequence modeling architectures. Further, we

provide a comprehensive, consistent and reproducible empirical evaluation of Transformer, Mamba and xLSTM within an SSAST-style masked spectrogram modeling framework on a suite of 10 varied utterance level audio classification tasks, based on our previous research [20], [44], allowing the reader to evaluate which sequence modeling approach best fits their needs. In addition to our previous work, we also evaluate how the covered sequence modeling approaches fare with different input sequence lengths and input audio clip durations.

Throughout the paper, we also discuss current challenges and possible future research directions. We also provide direct links to implementations and demos wherever possible, as well as source code(s) and pretrained models that can be used to reproduce the results presented in this study.

The rest of the paper is organised as follows. Section II covers MSMs and relevant concepts and components. Section III provides a comprehensive overview of Mamba and xLSTM, the two neural sequence modeling architectures in spotlight. Section IV covers the SSAST-style masked spectrogram modeling framework under which we investigate Transformer, Mamba and xLSTM based MSMs, as well as commentary on datasets and metrics used for the study. This is followed by Section V where we discuss the outcomes of the empirical evaluation, along with select ablation studies. Section VI discusses the things that we could not cover in the presented analysis. Finally, Section VII presents our concluding remarks.

II. TRANSFORMER BASED MASKED SPECTROGRAM MODELING

Masked language models such as BERT [1] learn strong semantic language representations without supervision by learning to predict masked portions of tokenized input text using visible, unmasked portions. Masked spectrogram models (MSMs) operate on the same underlying principles. The input audio spectrogram is divided into (generally) non-overlapping patches, and a portion of these patches is then randomly masked. MSMs learn contextualized audio representations by forcing them to reconstruct masked out patches based on the visible ones. Upcoming is a comprehensive discussion of the core components in masked spectrogram modeling, along with commentary on how these components differ across SSAST and MAE style MSMs, wherever applicable. An overview of MSMs can be found in Figure 1.

A. Creating and embedding audio spectrogram patches

The first step is to create non-overlapping audio spectrogram patches from an input audio spectrogram. The reasoning behind computing non-overlapping patches is to avoid information leakage across different patches, which would make the reconstruction task easier and result in worse performance, as demonstrated by [17].

Let $\mathbf{X} \in \mathbb{R}^{T \times F}$ denote the input audio spectrogram with T frames and F frequency bins. Then, we extract N non-overlapping patches of shape $t \times f$ along the time and frequency axes, respectively, and flatten them to yield patched input spectrograms $\mathbf{X}_p \in \mathbb{R}^{N \times t \cdot f}$. Thus, while \mathbf{X}_p is a patched representation, it is worth noting that the representation space

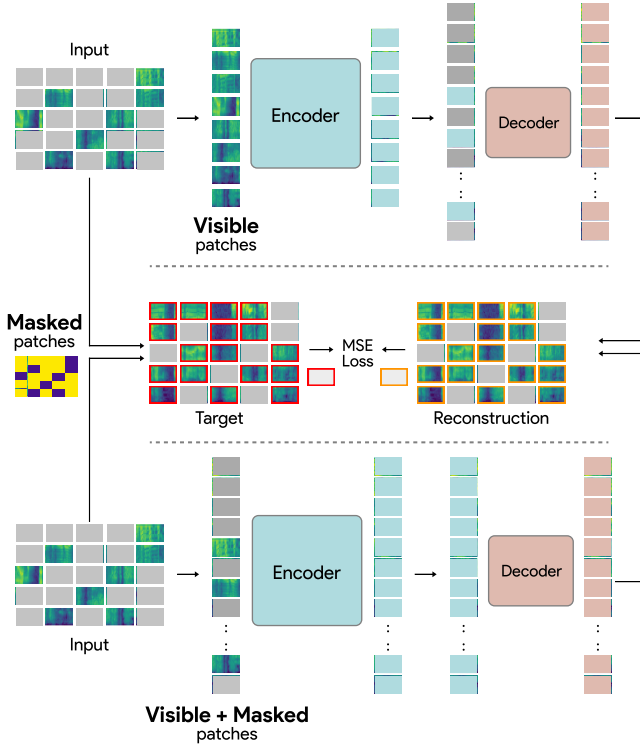


Fig. 1. The main types of masked spectrogram models: MAE-style (top), where the encoder operates only on the visible patches, and SSAST-style (bottom), where masked patches are replaced by learnable mask tokens before feeding to the encoder.

has not changed: they are essentially rearranged spectrograms. A learnable linear projection embeds them to a d_{enc} dimensional space, which yields patched input $\mathbf{X}_e \in \mathbb{R}^{N \times d_{\text{enc}}}$ ready for further processing. We can now optionally append a representative learnable class token $\text{cls} \in \mathbb{R}^{d_{\text{enc}}}$ to the beginning of \mathbf{X}_e . While not essential, the cls token acts like a *register* for the Transformer blocks to store global information in [46] and facilitate straightforward downstream fine-tuning of the MSM. Register tokens have also been shown to enable interpretable attention maps in Transformers [46]. For the sake of clarity and consistency, we omit the cls token in any equations and notation. Finally, to encode positional information, positional embeddings are added to \mathbf{X}_e . The vast majority of MSMs use fixed sinusoidal positional embeddings [16]–[20], [44]. However, if a different input duration is expected during test-time, appropriate measures, such as positional embedding extrapolation, or a better suited positional embedding, like Rotary Positional Embeddings (RoPE) [47], can be used.

Impact of patch size: The shape of the patch, $t \times f$, is an important hyperparameter and needs to be optimized together with the spectrogram computation parameters. Selection of t and f dictates the time-frequency resolution of the resulting patches and, effectively, the entire MSM, and the smaller the value, the finer the resolution. This allows MSMs to implicitly model both temporal and frequency structure from the input spectrogram, in contrast to several masked predictive modeling based speech representations, such as [10], [11], [13], [48], [49] where masking is applied only along the temporal axis.

Let us consider some common cases:

- 1) $f < F, t < T, f = t$: A square patch, which is quite prevalent and is adopted by a large number of approaches [16], [17]. For a given spectrogram, this results in a neutral setting without trading off time or frequency resolution for the other.
- 2) $f < F, t < T, f \neq t$: Several works have evaluated rectangular patches that offer different time-frequency resolution tradeoffs [18]–[20], [44]. Most notably, [18] conducted in-depth analysis with different patch resolutions, and demonstrate that patches with finer temporal resolutions performed better for speech based tasks.
- 3) $f = F, t = 1$: In this case, there is a patch for every frame in your input spectrogram, and every patch embeds frequency information from all the frequency bins. This setting is maximizing time resolution at a tradeoff for frequency resolution, and has been shown to work well for speech based applications, such as Keyword Spotting [50]–[52].

B. Masking strategies

As previously stated, in the same vein as MLMs like BERT [1], a portion of the embedded patches (\mathbf{X}_e) will be masked out. In contrast to MLMs, where masking between 10 – 20% of the input tokens is sufficient to learn good representations, MSMs have very high masking ratios (50-80%) similar to their image counterparts [7], [8]. In contrast to language, which is very information dense, spectrograms (especially high resolution spectrograms), can be highly spatially redundant. As a result, recovering missing patches is easier in the case of MSMs, and such a high portion of random masking is necessary to learn good representations. From here on, we will refer to masking ratio as $m_r \in (0, 1)$, which denotes the proportion of patches to be masked.

In literature, patch masking methodologies can be broadly classified into 3 categories:

- 1) Unstructured random masking, which is truly random masking of individual patches.
- 2) Structured random masking, where the mask follows a shape or structure. Examples include horizontal/vertical strips [18] and block-wise or grid masking, where a span of contiguous blocks or a grid are masked out.
- 3) Guided masking strategies, where the objective is to find optimal masking strategies based on the input. These include masking based on semantic concepts to hide [53] as well as masking based on the emerging concepts in the model itself [54], to name a few.

Most MSMs in literature utilize an unstructured random masking strategy, where tokens are masked out completely at random. This strikes the best balance between computational overhead, ease of implementation as well as performance on a varied bunch of tasks. However, selecting a structured or refined masking strategy based on domain-specific information for a particular application can pay off. Finally, finding optimal model or data informed masking strategies, such as using adversarial learning to determine which patches to mask for a given input, can be a potential avenue for future research.

C. Encoding contextual representations

After creating and embedding patches as well as computing the mask indices, it is time to feed the patches to the encoder, which is a stack of Transformer blocks of d_{enc} dimensions. Since the encoder is operating on a partial view of the input spectrogram (patches), it needs to learn high-level semantic information from the limited view and encode it into its outputs, so that the decoder can then reconstruct the missing input from them. The structure of the encoder is one of the key differences between SSAST-style and MAE-style MSMs. In SSAST-style models, the spectrogram patches to be masked, denoted by the stored mask indices, are replaced in-place by a learnable *mask token*, i.e. $\mathbf{X}_e \in \mathbb{R}^{N \times d_{\text{enc}}} \mapsto \mathbf{X}_{\text{enc}} \in \mathbb{R}^{N \times d_{\text{enc}}}$, where \mathbf{X}_{enc} denotes the input to the encoder. In contrast, in MAE-style models, the patches to be masked are *removed* from the input sequence using an index operation, i.e. $\mathbf{X}_e \in \mathbb{R}^{N \times d_{\text{enc}}} \mapsto \mathbf{X}_{\text{enc}} \in \mathbb{R}^{(1-m_r)N \times d_{\text{enc}}}$, where m_r is the masking ratio. As a result, the encoder in an MAE-style MSM ingests only a small fraction of the total number of input patches, allowing the usage of much larger encoders compared to an SSAST-style MSM for the same number of multiply-accumulate operations (MACs). This is made possible by the Transformer blocks, where the underlying scaled dot-product attention operation do not treat the input as a sequence, but rather as elements in a set of inputs with the computed attention weights measuring relative importance of each element with respect to every other. Thus, the MAE encoder is closely coupled with the Transformer, and is fundamentally incompatible with other sequence modelling architectures. Thus, the encoder returns $\mathbf{Z}_{\text{enc}} = \text{Encoder}(\mathbf{X}_e)$, where $\mathbf{Z}_{\text{enc}} \in \mathbb{R}^{N \times d_{\text{enc}}}$ for SSAST-style models and $\mathbf{Z}_{\text{enc}} \in \mathbb{R}^{(1-m_r)N \times d_{\text{enc}}}$ for MAE-style models.

The majority of the MSMs incorporate encoders comprised of standard Transformer layers with global self-attention, which might be suboptimal for audio recognition tasks where discriminative information is predominantly local. As a result, several works have evaluated MSMs with encoders based on modified Transformer blocks, such as the Swin Transformer [55], which can explicitly capture local and global information [17], [19]. This is also a noteworthy avenue for potential future research, since different audio recognition tasks operate at a different interplay of local-global dynamics. For instance, a hybrid, adaptive attention formulation might be able to give us the best of both worlds.

D. Reconstruction and loss computation

Once the visible patches have been embedded into \mathbf{Z}_{enc} , the decoder's task is to reconstruct the original spectrogram patches \mathbf{X}_p corresponding to the masked indices based on the high-level contextual information encoded in \mathbf{Z}_{enc} by the encoder. First, we linearly project \mathbf{Z}_{enc} into a d_{dec} dimensional space. In MAE-style MSMs, learnable *mask tokens* are then inserted in a manner that restores the correct order of the masked-unmasked positions, followed by injecting positional information again into the resulting vector using a positional embedding. In contrast, this operation is not needed for SSAST-style MSMs, since mask tokens were inserted prior to encoding and thus the \mathbf{Z}_{enc} representation is complete. At this

point, the number of tokens in both \mathbf{X}_p and the intermediate representation $\mathbf{Y}_d \in \mathbb{R}^{N \times d_{\text{dec}}}$ is N . Now, we need to project \mathbf{Y}_d to $t.f$ dimensions so that we can compute reconstruction loss.

In the case of SSAST-style MSMs, this is generally done using a linear layer with $(t.f)$ output features. In contrast, in the literature MAE-style MSMs usually include a stack of Transformer blocks with d_{dec} -dimensions before the final linear layer projecting to $t.f$. However, the decoder Transformer does not need to be as complex as the encoder, and can often have a fraction of the layers, as shown in [18], [19]. Some recent papers have even shown that there is no need for a Transformer-based decoder, and a convolutional decoder can be used instead [15], [56]. Either way, it is clear that due to their design, the decoding is more nuanced in the case of a MAE-style MSM.

The decoder yields the final reconstructed output $\mathbf{Y}' = \text{Decoder}(\mathbf{Z}_{\text{enc}})$ such that $\mathbf{Y}' \in \mathbb{R}^{N \times t.f}$, which has compatible shapes with \mathbf{X}_p . The objective function for pretraining the MSM is the mean squared error (MSE) between the *masked* spectrogram patches \mathbf{X}_p and the corresponding reconstructions \mathbf{Y}' , which has shown better performance versus computing MSE for all the patches [7], [16], [17]. While most MSMs use only a reconstruction objective, some papers also adopt auxiliary loss functions. For instance, [16], [21] also use an InfoNCE loss function to create a dual discriminative + reconstructive objective function, while other works introduce a contrastive learning loss function as an auxiliary objective [16], [57]. Furthermore, recent approaches like [58] completely change the target task from regression to a classification objective with self-distilled acoustic units as labels. Thus, while reconstruction alone yields good performance, looking at auxiliary loss functions that suit ideal representation characteristics can be useful, and is an interesting avenue for future research.

III. SEQUENCE MODELING BEYOND TRANSFORMERS

A. On Transformers and Scaled Dot-Product Attention

Transformers have emerged as the eminent neural architecture. Starting out as a new sequence modeling paradigm for NLP applications, Transformers have seen quick and widespread adoption across different input modalities and application domains. While there is a cause-and-affect relationship between emergent phenomena across data modalities, such as the concept of a patch as the atomic input unit in computer vision and audio, the prime drivers of this adoption are the strong generalization abilities and the inherently modality agnostic design of Transformers, including the scaled dot-product attention (SDPA) operation at their core. However, Transformers also have several drawbacks.

Quadratic Complexity: Recall that the following equation can be used to describe the self-attention operation:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}, \quad (1)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d_k}$. Here, the matrix multiplication for computing attention weights $\mathbf{Q} \cdot \mathbf{K}^T$ requires $N \times N$ operations and storage, and thus has quadratic compute and space

complexity with respect to sequence length N , which scales poorly for very large N . Furthermore, retrieval operations for a new query (\mathbf{Q}) requires storing the entire Key-Value (KV) cache.

Standard attention is global: As evident from Eq 1, attention computation is a global operation, as the process of attention matrix computation takes all N elements of K and Q into consideration. While this has the benefit of computing global long-range dependencies in the input, such an operation is inefficient for tasks that comprise predominantly of local information, such as speech. This is evidenced by the fact that compared to the standard Transformer, models that explicitly employ techniques that excel at modeling local information, such as convolutional layers, excel in speech recognition (e.g. Conformer [59]), and in other applications such as visual object detection and localization, where capturing fine-grained information is critical.

A lot of research efforts have been spent on addressing these shortcomings. Several papers attempt to address the quadratic complexity of Transformers by proposing sub-quadratic or linear approximations for SDPA [23], [24], [60], [61]. A lot of hardware aware implementations that improve real-world performance characteristics of Transformers, such as *FlashAttention* [25], [26] have also been proposed. Other methods, such as Multi-query attention (MQA) [62] and Grouped-query attention (GQA) [63] attempt to reduce the number of processed key-value pairs in order to improve runtime performance. Methods such as Ring Attention [64] attempt to address the high memory requirements of SDPA and enable scaling Transformers to very large context lengths, whereas hybrid local-global architectures offer better theoretical complexity [55] as well as improved local and hierarchical feature modeling [55], [59] capabilities. Finally, several approaches attempt to completely replace standard attention with other alternatives [27], [28], [65].

Needless to say, improving upon Transformers and SDPA is an area with incredibly high research interest. While there have been a lot of methods that attempt to address these issues, two recent independent lines of research have showed a lot of promise, namely selective structured state space models (a.k.a. Mamba) [36], [37] and extended long short term memory (xLSTM) [30], both of which are recurrent neural networks. We discuss these two methods in more detail in the following sections.

B. Selective Structured State Space Models

Recently, [29] proposed structured state space sequence models (S4) that excel at modeling long range dependencies in very long sequences. S4 models are related to CNNs, RNNs and classical state space models, and are based on a simple continuous system (Eq. 2).

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t) \quad (2a)$$

$$y(t) = \mathbf{C}h(t) \quad (2b)$$

S4 models are linear time-invariant (LTI) models governed by four parameters $(\Delta, \mathbf{A}, \mathbf{B}, \mathbf{C})$, and constitute a sequence-to-sequence transformation on input $\mathbf{x} \in \mathbb{R}^T$ through a latent space of dimensionality N that is comprised of two stages:

1. *Discretization stage:* which involves zero-order hold (ZOH) discretization of parameters \mathbf{A} and \mathbf{B} (Eq. 3)

$$\bar{\mathbf{A}} = \exp(\Delta\mathbf{A}) \quad (3a)$$

$$\bar{\mathbf{B}} = (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I})\Delta\mathbf{B} \quad (3b)$$

2. *Computation stage:* The model $\text{SSM}(\bar{\mathbf{A}}, \bar{\mathbf{B}}, \mathbf{C})(\cdot)$ can then be computed as a linear recurrence (Eq. 4a-4b) or as a global convolution (Eq. 4c-4d).

$$h_t = \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t \quad (4a)$$

$$y_t = \mathbf{C}h_t \quad (4b)$$

$$\bar{\mathbf{K}} = (\mathbf{C}\bar{\mathbf{B}}, \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}, \dots, \mathbf{C}\bar{\mathbf{A}}^{M-1}\bar{\mathbf{B}}) \quad (4c)$$

$$\mathbf{y} = \mathbf{x} * \bar{\mathbf{K}} \quad (4d)$$

Efficient computation of an S4 model necessitates that a structure be imposed on the transition matrix \mathbf{A} since random initialization leads to very poor performance, and thus the name *structured* state space models. The most common form of structure is diagonal, where \mathbf{A} is initialized using the *HiPPO* theory of continuous time memorization [66]. In equations (3)-(4), matrices $\mathbf{A} \in \mathbb{R}^{N \times N}$, $\mathbf{B} \in \mathbb{R}^{N \times 1}$ and $\mathbf{C} \in \mathbb{R}^{1 \times N}$ can all be represented by N numbers. Furthermore, given that it is an LTI system, the parameters $(\Delta, \mathbf{A}, \mathbf{B}, \mathbf{C})$ of S4 are fixed for all time-steps. The dual nature of S4 that enables computation as a structured global convolution (Eq. 4d) with a structured kernel $\bar{\mathbf{K}}$ facilitates efficient training, whereas computation via recurrence (Eq. 4b) enables efficient inference, thus offering a “best of both worlds”.

While S4 demonstrated excellent long-term dependency modeling capabilities and achieved state-of-the-art performance on the Long Range Arena [67] benchmarks, it has several drawbacks. S4, being an LTI system, is not content-aware. More specifically, S4 can neither select correct information from context nor adapt the hidden state based on information presented in a sequence, as evident from the transitions in Eq. 4. In simpler words: once the transition matrices are learned and hence fixed, during inference for a given input, there is no mechanism in S4 to retrieve relevant information for current time-step t based on previous time-step(s), nor is there a mechanism to reflect a relevant change in the hidden state. This is highlighted by empirical evidence provided by [36] that suggests that LTI SSMs struggle at selective copying and context-aware reasoning. Further, when operating on a B sized batch of D -channel sequences of length L , an SSM is to be applied *independently* on each channel. This is computationally inefficient as it requires applying a total hidden state that has DN dimensions per input and takes $O(BLDN)$ time and memory.

To address these issues, [36] proposed incorporating a selection mechanism to structured state space models, yielding the “selective structured state space sequence models with a scan” algorithm, also known as S6. S6 trades off time-invariance and equivalence to convolutions for input-dependent processing. The fundamental premise behind S6 is to make SSM parameters Δ, \mathbf{B} and \mathbf{C} functions of the input through

learned linear projections, thus adding an extra *time* dimension to each of these parameters, as depicted in Eq 5.

$$\mathbf{B} = s_B(x) \quad (5a)$$

$$\mathbf{C} = s_C(x) \quad (5b)$$

$$\Delta = \text{softplus}(\Delta_p + s_\Delta(x)) \quad (5c)$$

where, $s_B(\cdot)$ and $s_C(\cdot)$ are N dimensional parameterized linear projections, whereas s_Δ is a linear projection to unit dimension broadcasted to all D channels. Δ_p symbolizes the actual learnable parameter underpinning Δ . Thus, effectively, for a batch of B input sequences of length L and channels D , parameters \mathbf{B} and \mathbf{C} are tensors of shape $(B \times L \times N)$, Δ is a tensor of shape $(B \times L \times D)$ and the resulting discretized parameters $\bar{\mathbf{A}}, \bar{\mathbf{B}}$ are tensors of shape $(B \times L \times D \times N)$. Finally, the output can be computed using the same $\text{SSM}(\bar{\mathbf{A}}, \bar{\mathbf{B}}, \mathbf{C})(\cdot)$ formulation as presented earlier, however, since the equivalence to convolution has been lost due to the tensor shapes and input-dependence, it can be computed only in the recurrence mode (Eq. 4a-4b). Intuitively, one could look at the S6 architecture, and its constituent parameters and elements, as follows:

- The selection mechanism in S6 is closely related to classical RNNs, with the Δ parameter playing the role of a generalized RNN gating mechanism. Δ parameter governs the persistence of hidden state h when presented with input \mathbf{x} : a large value of Δ resets h and focuses on the contents of the current input x_t , whereas a smaller Δ results in persistence of the state.
- It is worth noting that the parameter \mathbf{A} is not selective. The reasoning is that it does not need to be, since the model only interacts with \mathbf{A} through $\bar{\mathbf{A}}$, which in turn is impacted by the selective parameter Δ during the discretization process.
- Parameters \mathbf{B} and \mathbf{C} can be further viewed as gates that affect how input x_t impacts the hidden state h (\mathbf{B}) and how the hidden state h impacts the output y_t (\mathbf{C}).

In the case of S6, the key bottleneck from an implementation standpoint is the complexity of the recurrence computation ($O(BLDN)$). While a parallel scan algorithm can be used to implement the recurrence, together with kernel fusion techniques, efficient materializing of the two discretized input tensors $\bar{\mathbf{A}}, \bar{\mathbf{B}}$ and the hidden state h is necessary, since operations on these tensors have a considerable memory bandwidth cost. To facilitate this, a parallel scan implementation that is aware of the accelerators' memory hierarchy, such that discretization and recurrence occurs directly in faster memory levels of the accelerator (SRAM), with final outputs written directly to slower but larger memory levels (HBM), is developed. Finally, [36] propose a simplified plug-and-play SSM building block, called *Mamba* (Figure 2), which is a combination of an H3 block [32] and a gated MLP block quite prevalent in modern DNNs. From hereon, we use the term *Mamba* to refer to both the Mamba blocks and/or S6, depending on the context. Thus, most existing neural architectures can be fitted with an SSM by simply replacing Transformer blocks with Mamba block(s) of appropriate dimensions. Mamba scales linearly with input sequence length, and offers up to 5x

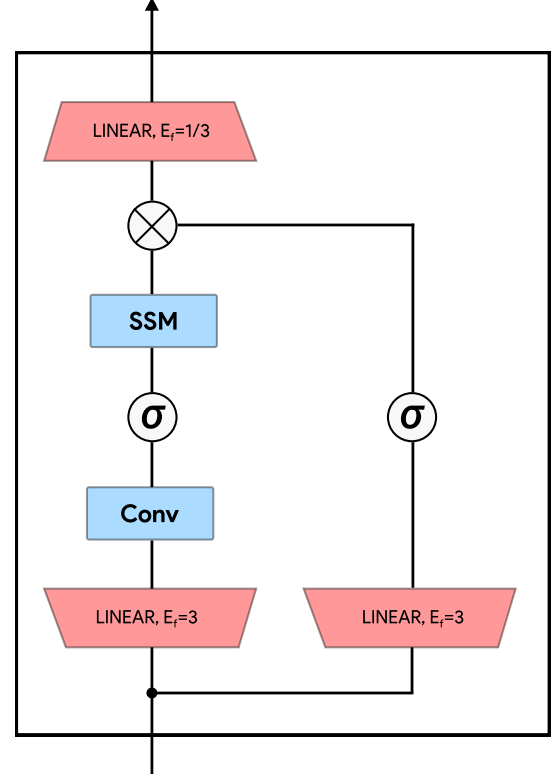


Fig. 2. The Mamba block, where SSM represents the Mamba SSM formulation and σ signifies the Swish activations. Red blocks are linear projections.

better inference throughput compared to Transformers [36]. It can also perform well on up to a million-length sequences, outperforming comparable Transformer models for language modeling, audio and genomics, as demonstrated by [36].

Consequently, Mamba has gained significant traction across several domains since its release, such as visual recognition [68], [69], object recognition and detection [70], video understanding [71], point-cloud analysis [72], [73], DNA sequence modeling [74], biomedical image segmentation [75], language modeling [76]–[78] as well as autoregressive pretraining for foundation models for vision [79], [80]. Several derivatives of Mamba have also been proposed, such as Hydra [81], which proposes generalized matrix mixers for bidirectional state space modeling, and LocalMamba [82], which proposes a windowed selective scan algorithm for better modeling of local dependencies. In the speech and audio processing domain, Mamba has been adopted for speech recognition [38], [83]–[87], speech enhancement [88]–[93], speech separation and speaker extraction [39], [94]–[96], depression detection [97]–[99], speech super-resolution [100], emotion recognition [101], music and sound generation [102]–[104], deepfake detection [105], [106], audio representation learning and classification [41], [42], [107], [108], and finally, in SSAST-style MSMs for learning self-supervised audio representations [20], [40], [109].

However, Mamba, or other selective state space models for that matter, are not without drawbacks. For instance, Mamba is intrinsically a unidirectional model, which is suboptimal for

several tasks and domains. Several works have attempted to address this issue, either by proposing Mamba blocks with additional SSM branch(es) that operate on a flipped version of the input [69], [83], using multi-dimensional parallel scan operations [68], [110], or by matrix mixing [81]. Using a hybrid Transformer-Mamba approach is also very popular [78], [91], [111]. It is fair to say that Mamba and state space models have amassed tremendous interest from the community at large. Our understanding of Transformers comes from years of research in the field, and with several challenges yet to be solved and properties yet to be discovered, we believe that state space models such as Mamba are to become a mainstay in signal processing and machine learning.

C. Extended long short-term memory

Before the advent of Transformers and state-space models, LSTMs [43] were the go to neural architecture for sequence modeling and powered several key breakthroughs in neural machine translation [112], [113], automatic speech recognition [114], [115], image captioning [116], [117], image generation [118] and audio synthesis [119], to name a few. Transformers were able to outperform LSTMs and become the de facto sequence modeling architecture in lieu to the following shortcomings of the latter:

1. *Compression into a scalar cell state:* At the heart of the LSTM is a scalar memory cell, which interacts with three gates: input, output and forget gate. All information processed by the LSTM thus has to be compressed and stored in this scalar memory cell, which limits its storage capacity. This can lead to several issues, one of which is poor performance on rare input tokens compared to Transformers, which have no compression of context at all.

2. *Sigmoid gating and inability to revise storage decisions:* In LSTM, the input, output and forget gates have a sigmoid activation for gating. Since sigmoid squashes the input into the $[0, 1]$ range in an S-shaped curve, very large and very small values at the proximities are flattened to values very close to each other. This flattening effect can impact fine-grained decision making regarding storage, and thus LSTM can struggle to revise a stored value when a more similar vector is found.

3. *Hidden-hidden connections and lack of parallelizability:* In LSTMs, hidden-hidden connections between hidden state from one time step to the next, which is also referred to as *memory mixing*, mandates sequential processing. Despite the constant memory and linear computation complexity of the LSTM, in practice this is a bottleneck in training with hardware acceleration. On the other hand, while the SDPA operation has quadratic compute and memory complexity, the inherently parallelizable nature of the SDPA operation facilitates fast training and better scalability, which was a big win for the Transformer architecture.

The Extended Long Short-Term Memory, a.k.a. xLSTM architecture, [30] addresses these issues and proposes two core memory cell blocks: sLSTM and mLSTM, both of which incorporate exponential gating on the input and forget gates to enhance storage decision revision capabilities, along with new normalizer and stabilizer states for better training stability.

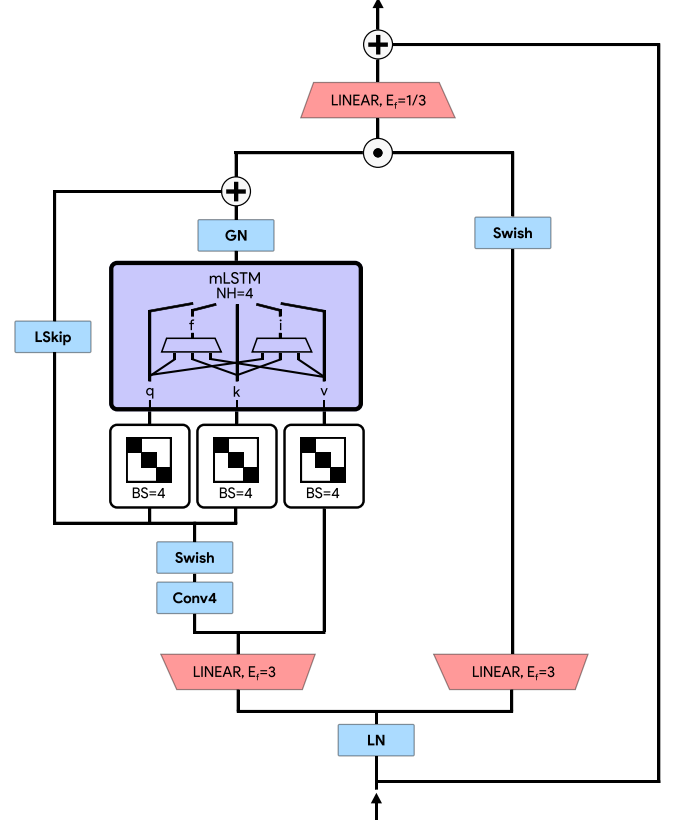


Fig. 3. The ViL block fitted with the mLSTM module. GN, LN stand for GroupNorm and LayerNorm, respectively. Red blocks are linear projections.

sLSTM introduces the concept of *heads*, similar to the Transformers, where multiple memory cells can be grouped together in different heads. A normalizer state is introduced, which sums up the product of the input gate times all future forget gates. The scalar sLSTM forward pass is as follows:

$$c_t = f_t c_{t-1} + i_t z_t \quad (6a)$$

$$n_t = f_t n_{t-1} + i_t \quad (6b)$$

$$h_t = o_t \tilde{h}_t, \quad \tilde{h}_t = c_t / n_t \quad (6c)$$

$$z_t = \tanh(\mathbf{w}_z^t \mathbf{x}_t + r_z h_{t-1} + b_z) \quad (6d)$$

$$i_t = \exp(\mathbf{w}_i^t \mathbf{x}_t + r_i h_{t-1} + b_i) \quad (6e)$$

$$f_t = \exp(\mathbf{w}_f^t \mathbf{x}_t + r_f h_{t-1} + b_f) \quad (6f)$$

$$o_t = \text{sigmoid}(\mathbf{w}_o^t \mathbf{x}_t + r_o h_{t-1} + b_o) \quad (6g)$$

where c_t is the cell state, n_t is the normalizer state, h_t is the hidden states, z_t is the cell input and i_t, f_t, o_t represent the input, forget and the output gate, respectively. Weight vectors between the input \mathbf{x}_t and the corresponding unit/gate are represented as \mathbf{w}_* , whereas recurrent weights between hidden state h_{t-1} and the corresponding unit/gate are represented as r_* . b_* represent bias terms. Since exponential activations ($\exp(\cdot)$) can lead to large values that can cause overflows,

input and forget gates are stabilized as follows:

$$m_t = \max(\log(f_t) + m_{t-1}, \log(i_t)) \quad (7a)$$

$$i'_t = \exp(\log(i_t) - m_t) \quad (7b)$$

$$f'_t = \exp(\log(f_t) + m_{t-1} - m_t) \quad (7c)$$

where m_t denotes the stabilizer state. In lieu to a new memory mixing strategy that enables inter-head memory mixing, i.e., hidden-hidden connections between time steps within the constituent memory cells in a head, but no intra-head mixing, sLSTM allow sequential-only processing, and are closer to the original LSTM.

mLSTM enhances the storage capacity of LSTMs by replacing the scalar memory cell $c \in \mathbb{R}$ in the LSTM with a matrix cell $\mathbf{C} \in \mathbb{R}^{d \times d}$. Eq. 8 depicts the forward pass for an mLSTM block for an input \mathbf{x}_t . Note that as opposed to sLSTM, states are vectors.

$$\mathbf{C}_t = f_t \mathbf{C}_{t-1} + i_t \mathbf{v}_t \mathbf{k}_t^T \quad (8a)$$

$$\mathbf{n}_t = f_t \mathbf{n}_{t-1} + i_t \mathbf{k}_t \quad (8b)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \left(\frac{\mathbf{C}_t \mathbf{q}_t}{\max(|\mathbf{n}_t^T \mathbf{q}_t|, 1)} \right) \quad (8c)$$

$$\mathbf{q}_t = \mathbf{W}_q \mathbf{x}_t + \mathbf{b}_q \quad (8d)$$

$$\mathbf{k}_t = \frac{1}{\sqrt{d}} \mathbf{W}_k \mathbf{x}_t + \mathbf{b}_k \quad (8e)$$

$$\mathbf{v}_t = \mathbf{W}_v \mathbf{x}_t + \mathbf{b}_v \quad (8f)$$

$$i_t = \exp(w_i^T x_t + b_i) \quad (8g)$$

$$f_t = \exp(w_f^T x_t + b_f) \quad (8h)$$

$$\mathbf{o}_t = \text{sigmoid}(\mathbf{W}_o \mathbf{x}_t + \mathbf{b}_o) \quad (8i)$$

Aligning with the terminology introduced by Transformers, mLSTMs computes query (\mathbf{q}), key (\mathbf{k}), and value (\mathbf{v}) vectors from the input vector \mathbf{x} , also introducing corresponding weight matrices ($\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$, respectively) and bias terms. The core memory cell \mathbf{C} is formed as the outer product between the key and the value vectors, abiding with the covariance update rule given that keys and values are expected to have zero mean since projection is proceeded by a layer-norm. In contrast to the sLSTM, multiple memory cells and heads are equivalent in an mLSTM since there is no memory mixing, and thus the mLSTM cell can be parallelized. mLSTM uses the same stabilization as sLSTM (Eq. 7). [30] propose two different residual blocks for the two types of cells. The sLSTM follows a *post-up projection* like Transformers, where sequence modeling is done in the original space followed by projection to a higher dimensional space and applying the non-linearity, whereas the mLSTM block adopts a *pre-up projection* block similar to Mamba where projection to a higher dimensional space is done before sequence modeling, facilitating a larger memory capacity. The xLSTM architecture, thus, is a stack of these sLSTM and/or mLSTM blocks. xLSTMs have demonstrated excellent performance for language modeling, performing on par or better than Transformers [30], [120]. While being more recent compared to state space models, xLSTMs have been leveraged in a variety of domains, such as ViL, a patch based image recognition approach similar to Vision Transformers [121] which leverages only the mLSTM blocks, as well

as generative and in-context learning of DNA, protein and chemical sequences [122]. xLSTMs are also seeing active adoption in remote sensing [123], time-series modeling [124]–[126] and medical image segmentation [127]–[130], majority of latter building on top of ViL blocks. In the audio and signal processing domain, xLSTMs have being used for speech enhancement [45], and finally, in SSAST-style MSMs for learning self-supervised audio representation [44], which also uses ViL based blocks that comprise primarily of mLSTM cells. Thus, it is evident that between the two proposed blocks, the mLSTM block has definitely garnered much more attention, and is utilized in upcoming sections of this paper. Figure 3 depicts the mLSTM based ViL block.

The xLSTM architecture has it's own set of flaws. Until very recently, xLSTMs were much slower in practice compared to Transformers, despite their linear complexity with respect to input sequence lengths. This was recently addressed in [131], where more performant GPU kernels were proposed. Similar to Mamba, xLSTMs are also intrinsically a unidirectional model, and requires alternating blocks that operate on flipped input for bidirectional modeling [121]. Further, from practical real world usage, xLSTMs have been known to require more memory compared to Transformers for short length sequences. Nevertheless, despite being a more recent development than Mamba, xLSTMs are seeing active adoption in the field, and are a promising research direction in signal processing and machine learning.

In the upcoming sections we aim to provide an empirical comparison of Transformers, Mamba and xLSTMs for masked spectrogram modeling across a wide variety of downstream audio recognition tasks within a consistent, reproducible unified MSM framework.

IV. APPROACH

To evaluate Transformers, Mamba and xLSTM under a unified framework, we adopt an SSAST-style MSM. We chose SSAST over MAE for a critical reason: encoding as done in MAE is closely coupled and only possible with the Transformer architecture because it treats input as a set of tokens and assumes no sequential relationship between consecutive patches. Picking a framework that is too closely tied with a particular sequence modelling approach prohibits evaluation of alternative, upcoming approaches. MAE is inherently incompatible with Mamba and xLSTM, which are fundamentally recurrent neural networks that are explicitly modelling sequential dynamics in the input. In contrast, the SSAST design is not coupled with the Transformer architecture, and can be used to evaluate all kinds of sequence modelling approaches. Thus, the only way to conduct an unbiased comparison of Transformers with Mamba and xLSTM without additional confounding factors was to do so within the SSAST framework. SSAST was the first patch-based self-supervised learning framework for audio spectrogram Transformers, and it is a well established baseline that will facilitate comparisons across a multitude of application domains. In the upcoming section we first discuss the datasets used in the study, followed by a description of the unified MSM framework used for comparison. Finally, we discuss the post-pretraining downstream evaluation setup.

A. Datasets

Pretraining: We use the AudioSet dataset [132] for self-supervised pretraining. AudioSet is one of the largest audio corpus available, with roughly 2-million 10-second long weakly labeled YouTube video clips that span an exhaustive hierarchy of over 527 labels, totalling over 5000 hours of audio data. The choice for AudioSet was straightforward: it is a prominent dataset widely used for training audio representations [133]–[135], and almost all popular MSMs in literature have used AudioSet [16]–[19].

Downstream Evaluation: The objective of self-supervised pretraining for audio is to create a general purpose audio representation that performs well across a wide variety of audio recognition tasks. There are innumerable well-established audio recognition datasets in existence across a wide variety of domains, such as keyword spotting [136], environmental audio classification [137], audio tagging [132], [138] and emotion recognition [139], [140], to name a few. Furthermore, several benchmark suites have emerged to evaluate the performance of audio representations on a wide swath of datasets, such as SUPERB [141] for evaluating speech representations. These benchmark suites consolidate datasets that cover a wide variety of downstream tasks and provide standardized and reproducible evaluation protocols. HEAR [142] is one such suite. HEAR covers over 19 downstream audio recognition tasks, including a mix of established prominent audio recognition benchmarks as well as new ones, and has been used by several MSMs for evaluation [18], [19], [22]. However, several of these tasks are either redundant or highly correlated in performance [142]. Thus, similar to [19], we utilize a 10-task subset of the HEAR benchmark, for evaluating our MSM, which offers the best tradeoff between sufficient variety in downstream tasks while limiting the number of evaluations conducted. More specifically, for evaluating music and pitch perception characteristics of the evaluated MSMs, we use Beijing Opera [142], [143], Mridangam Stroke and Tonic [144] and the NSynth Pitch-5h [142], [145] tasks. Further, we use Crema-D [139], Speech Commands 5h [136], [142], LibriCount [146] and VoxLingua107 [147] to evaluate MSM performance on speech based tasks. Finally, we use ESC-50 [137] and FSD50K [138] to evaluate general audio classification performance. More information about the downstream tasks can be found in Table I, and in their respective references.

TABLE I
DESCRIPTION OF THE EVALUATED DOWNSTREAM TASKS. #CLASSES AND #HOURS REPRESENT THE NUMBER OF CLASSES AND THE TOTAL DURATION IN HOURS, RESPECTIVELY.

ID	Name	Description	#Classes	#Hours
BO	Beijing Opera	percussion instrument classification	4	0.3
CD	Crema-D	emotion recognition	6	10
E50	ESC-50	environmental sound classification	50	2.77
LC	LibriCount	counting speakers (classification)	10	8
Mri-S	Mridangam Stroke	classifying Mridangam <i>strokes</i>	10	1.57
Mri-T	Mridangam Tonic	classifying Mridangam <i>tonics</i>	6	1.57
NS-5h	NSynth Pitch 5h	pitch classification	88	5.5
SC-5h	SpeechCommands 5h	keyword spotting	12	6.5
F50K	FSD50K	multilabel audio tagging	200	100
VL	VoxLingua107 Top10	spoken language identification	10	5

B. Model architecture

To compare Transformers, Mamba and xLSTM, we utilize an SSAST-MSM framework as specified in Section II, and then essentially replace the Transformer blocks with Mamba blocks as specified Section III-B and Figure 2 and xLSTM blocks as specified in Section III-C and Figure 3. The resulting models with Transformer, Mamba and xLSTM based sequential modeling are hereafter referred to as SSAST, Self-Supervised Audio Mamba (SSAM) [20] and Audio-xLSTMs (AxLSTM) [44], respectively. In this section, we elucidate the specifics of the adopted SSAST-style MSMs across the Transformer, Mamba and xLSTM sequence modeling paradigms.

Creating and masking patches: Following II-A and II-B, all the models follow identical patch creation and masking. Log-scaled mel spectrogram features extracted from audio clips sampled at 16000 Hz with a window size of 25 ms, a hop size of 10 ms and $F = 80$ mel-spaced frequency bins are used as inputs, and in our default configuration we use a 2-second random audio crop for self-supervised pretraining. Unless specified otherwise, we compute (4×16) shaped rectangular patches, i.e. our patch parameters are $t = 4, f = 16$. This corresponds to spectrogram input to the MSM $\mathbf{X} \in \mathbb{R}^{200 \times 80}$, which after creating $t = 4, f = 16$ sized patches yields $\mathbf{X}_p \in \mathbb{R}^{250 \times 64}$, which are projected to d_{enc} (covered in the next paragraph). Like majority of the MSMs, we use unstructured random masking, along with a masking ratio of $m_r = 0.5$ for all models, which is in line with [16]. We add a representative cls token for consistent comparison with previous works as well as ease of finetuning for future works that use our released models. A fixed sinusoidal embedding is used for encoding positional information.

Encoding and reconstruction: The encoder in our default configuration for all the models match up with the ViT-Tiny [148] specifications, i.e. a stack of $l = 12$ blocks, $d_{\text{enc}} = 192$ and number of attention heads, $h = 3$. We also evaluate Small ($d_{\text{enc}}=384, l=12, h=6$) and Base ($d_{\text{enc}}=768, l=12, h=12$) encoder configurations across the board. As evident from Figure 2, Mamba blocks apply the SSM on expanded input, before projecting it back to d_{enc} dimensions. Similarly, the mLSTM based ViL block also expands the input (Figure 3). For Mamba and mLSTM blocks, the default expansion factor is $E_f = 2$, which results in roughly every 2 Mamba/mLSTM blocks having similar number of parameters as a single Transformer block. However, using these settings as is would mean that all Mamba/xLSTM based models were 2 times deeper than the corresponding Transformer based model. To control for neural network depth, which has been shown to directly impact the modeling capabilities of neural networks [149], we use an expansion factor of $E_f = 3$ for both mLSTM-based ViL blocks and the Mamba block. Additionally, the Mamba block also uses larger internal dimensions ($d_{\text{state}} = 24, d_{\text{conv}} = 4$). These choices, while still resulting in Mamba/xLSTM based MSMs to have fewer parameters than their Transformer counterparts, allow us to close the gap while controlling for model depth. Furthermore, neither Mamba nor xLSTM based MSMs are bi-directional, as bi-directional processing did not improve performance as per ablations. After encoding, a single hidden

layer MLP is used for reconstruction, where the hidden layer has dimensions $d_{\text{dec}} = d_{\text{enc}}$ and the final layer has dimensions (t, f) , with a GELU non-linear activation after the hidden layer.

Optimization: As highlighted in Section II-D, the objective function for pretraining all the MSM is mean squared error (MSE) between the *masked* spectrogram patches and the reconstructions. This is the key difference between all the evaluated MSMs and the original SSAST [16] which utilized the InfoNCE loss function in addition to MSE. All the evaluated models are pretrained for a 100 epochs with a batch size of 1024 and a weight decay of 0.05 with the AdamW optimizer [150]. A linear warmup for 10 epochs followed by a cosine learning rate decay schedule is used. In accordance with several MSMs in literature [16]–[19], no data augmentations were used.

C. Downstream evaluation and performance metrics

As previously stated, the HEAR has a standardized and reproducible evaluation protocol, implemented and released in the public *hear-eval-kit*. We evaluate MSMs on downstream tasks in accordance with the HEAR protocol: we extract fixed-sized feature vectors independent of the input audio duration by taking the mean over time across 2-second audio chunks. Then, a single hidden layer MLP classifier with 1024 neurons is trained for each task, repeated 10 times for each evaluated model. We then report 95% confidence intervals based on these runs. This setting represents a restricted hyperparameter evaluation grid compared to the one used to obtain results in [142], which might cause some discrepancies in numbers reported for the same task between [142] and ours. However, this was necessary given the large number of experiments conducted, and is in line with existing work [19], [20], [44]). While these results differ from a full fine-tuning use case, this evaluation protocol results in a thorough qualitative assessment of the evaluated self-supervised audio representations, where fine-tuning performance would be more bound by the number of parameters of the model, and not to mention, incredibly inefficient and wasteful of compute resources.

Aggregated Performance Metric: While all the evaluated downstream tasks are utterance level classification tasks, they are all vary significantly in difficulty and *breadth* in the underlying objective. Furthermore, different self-supervised representations will naturally do better on some tasks versus the other. Additionally, keeping track of performance across all the tasks is arduous, especially for ablations. Thus, using an aggregated performance metric to quantify the performance of an evaluated self-supervised model across all the tasks, that also takes task complexity into account, would be incredibly useful. To this end, we use the aggregated normalized score as proposed by [19] to compare evaluated approaches across the proposed list of downstream tasks. For a model m , overall score $s(m) \in [0, 100]$ is given as:

$$s(m) = \frac{1}{|T|} \sum_{t \in T} \frac{x_t(m) - \min_t}{\max_t - \min_t} * 100 \quad (9)$$

where $x_t(m)$ denotes performance of the model m on task t , and \min_t and \max_t represent the worst and the best performance across all models on the task, thus taking into account

the relative performance amongst all evaluated representations. This metric is similar to the one proposed by the SUPERB leaderboard [141], with a difference in the scale and range of the metric.

V. EMPIRICAL ANALYSIS

A. Establishing contextual performance w.r.t. literature

While our primary objective is to compare Transformers, Mamba and xLSTM based MSMs against each other, it is necessary to contextualize their performance compared to representations in literature. To this end, alongside pretraining SSAST-style Transformer, Mamba and xLSTM MSMs, we also conducted downstream experiments on several prominent audio representations, as shown in Table II. As a supervised baseline, we used the Patchout faST Spectrogram Transformer (PaSST) [151]. Trained on the AudioSet data in a supervised manner, PaSST establishes a very strong audio recognition baseline, improving upon plain Audio Spectrogram Transformers (ASTs) [152] by a considerable margin. We also include prominent self-supervised speech representations: wav2vec 2.0 [10], HuBERT [11] and WavLM [13] in our analysis, since these approaches have seen widespread adoption and are a useful landmark to navigate self-supervised audio representation performance. Finally, we include several popular existing MSMs, including the original SSAST [16] and other MAE-style MSMs [17]–[19], [58], to better contextualize performance of evaluated MSMs.

In Table II, the directly comparable Transformer (SSAST, underlined), Mamba (SSAM) and xLSTM (AxLSTM) approaches can be found in the “ours” section, and these models have identical feature embedding sizes. Given that it is infeasible to retrain all existing pretrained representations to have the same embedding sizes, we had to use existing representations as is. While it might seem suboptimal, this is a well-established evaluation protocol for comparing self-supervised audio representations [141], [142]. It is worth noting that SSAST [16] represents the officially released model trained on AudioSet+LibriSpeech, whereas the underlined SSAST models represent the architecture highlighted in Section IV-B which was pretrained by us. As evident from the table, SSAST-based MSMs fitted with Mamba (SSAM) and xLSTM (AxLSTM) blocks outperform comparable Transformer based SSAST baselines by a considerable margin in overall score $s(m)$. For the evaluated SSAST-style MSMs, the general performance trend indicates that Mamba-based SSAMs $>$ xLSTM-based AxLSTMs $>>$ SSASTs. SSAMs consistently outperform SSASTs by roughly 30% in relative performance, whereas they are only 3 – 4% better than comparable AxLSTMs counterparts. Both SSAM-Base and AxLSTM-Base outperform MAE-style approaches that are based on the standard Transformer block like AudioMAE, BEATs-iter3 and MSM-MAE-208, while having fewer parameters. Only the MWMAE [19] models, which possess a modified, non-standard multi-head attention module explicitly designed to capture local-global attention are able to outperform SSAM and AxLSTM models, with MWMAE-Large achieving the best overall performance across all evaluated audio representations. While SSAMs outperform AxLSTMs slightly in

TABLE II

COMPARING EVALUATED MSMS WITH POPULAR SELF-SUPERVISED AUDIO REPRESENTATIONS. LS, AS, VP, LL STAND FOR LIBRISPEECH, AUDIOSET, VOXPOPULI AND LIBRILIGHT DATASETS, RESPECTIVELY. ORIGINAL SSAST [16] WAS TRAINED ON AS+LS, WHEREAS WE PRETRAINED THE DIRECTLY COMPARABLE UNDERLINED SSAST BASELINES (SSAST-TINY, SMALL, BASE). *INCLUDES DECODER PARAMETERS

Model	Data	#M Params	Music & Pitch				Speech-based tasks				Audio		
			BO	Mri-S	Mri-T	NS-5h	CD	LC	SC-5h	VL	E50	F50K	$s(m)$
Supervised Baselines													
HEAR-Naive [142]	-	-	52.6 \pm 2.4	38.0 \pm 1.3	36.4 \pm 1.9	18.6 \pm 4.4	30.9 \pm 0.8	33.5 \pm 1.1	8.5 \pm 0.4	11.2 \pm 0.5	5.8 \pm 0.2	7.1 \pm 0.2	5.1 \pm 0.7
PaSST-Base [151]	AS	86	94.9 \pm 0.5	96.5 \pm 0.1	87.6 \pm 0.6	23.3 \pm 0.9	61.0 \pm 0.3	60.1 \pm 0.2	66.6 \pm 1.4	25.5 \pm 0.8	94.8 \pm 0.3	64.2 \pm 0.1	73.7 \pm 0.4
SSL													
W2V2-base [10]	LS	94.4	74.0 \pm 1.0	77.3 \pm 0.2	55.1 \pm 0.3	7.4 \pm 0.8	46.4 \pm 0.3	51.2 \pm 0.2	90.8 \pm 0.3	35.5 \pm 0.8	31.1 \pm 0.4	18.1 \pm 0.1	43.2 \pm 0.2
W2V2-large [10]	VP	315.4	93.1 \pm 0.7	93.9 \pm 0.1	77.4 \pm 0.2	42.0 \pm 1.0	66.9 \pm 0.4	62.4 \pm 0.3	87.6 \pm 0.5	53.6 \pm 1.0	60.1 \pm 0.5	34.2 \pm 0.1	74.2 \pm 0.4
HuBERT-base [11]	LS	94.4	92.1 \pm 0.6	94.4 \pm 0.1	84.9 \pm 0.3	19.4 \pm 0.7	70.8 \pm 0.2	56.5 \pm 0.3	93.2 \pm 0.1	61.8 \pm 0.6	57.8 \pm 0.6	32.3 \pm 0.1	72.7 \pm 0.2
HuBERT-large [11]	LL	315.4	94.1 \pm 0.7	95.3 \pm 0.1	83.5 \pm 0.3	19.3 \pm 0.8	70.7 \pm 0.1	59.9 \pm 0.2	83.2 \pm 0.7	66.1 \pm 0.9	60.3 \pm 0.4	31.5 \pm 0.1	73.6 \pm 0.3
WavLM-base [13]	LS	94.4	89.4 \pm 0.7	95.1 \pm 0.1	83.4 \pm 0.2	37.3 \pm 0.8	56.3 \pm 0.2	63.2 \pm 0.3	57.2 \pm 0.8	22.6 \pm 0.6	46.6 \pm 0.4	29.9 \pm 0.1	60.7 \pm 0.2
WavLM-large [13]	Mix	315.4	96.4 \pm 0.5	96.8 \pm 0.1	89.5 \pm 0.1	53.7 \pm 0.5	57.2 \pm 0.2	61.1 \pm 0.3	46.2 \pm 0.8	23.7 \pm 0.9	47.9 \pm 0.4	29.0 \pm 0.1	64.2 \pm 0.2
MAE-Style MSMs													
AudioMAE [17]	AS	86.0	93.7 \pm 0.6	89.2 \pm 0.2	86.6 \pm 0.2	64.5 \pm 0.8	68.2 \pm 0.2	42.2 \pm 0.2	28.6 \pm 1.5	29.7 \pm 1.0	60.6 \pm 0.4	37.9 \pm 0.1	63.1 \pm 0.3
MSM-MAE-208 [18]	AS	92.7*	95.7 \pm 0.7	97.3 \pm 0.1	97.9 \pm 0.1	69.1 \pm 0.5	68.7 \pm 0.2	63.8 \pm 0.5	85.7 \pm 0.3	40.3 \pm 0.6	78.4 \pm 0.6	49.5 \pm 0.1	85.3 \pm 0.2
MWMAE-Tiny [19]	AS	12.6*	93.3 \pm 1.0	97.1 \pm 0.1	97.6 \pm 0.1	68.1 \pm 0.4	64.4 \pm 0.2	65.5 \pm 0.3	77.0 \pm 0.6	28.6 \pm 1.1	71.9 \pm 0.5	43.4 \pm 0.1	79.3 \pm 0.3
MWMAE-Base [19]	AS	92.5*	96.0 \pm 0.5	97.4 \pm 0.1	97.9 \pm 0.1	69.3 \pm 0.6	73.1 \pm 0.3	68.8 \pm 0.2	90.9 \pm 0.2	44.2 \pm 0.9	81.2 \pm 0.4	51.2 \pm 0.2	89.4 \pm 0.2
MWMAE-Large [19]	AS	308.9*	95.9 \pm 0.5	97.4 \pm 0.0	98.2 \pm 0.1	71.2 \pm 0.7	76.1 \pm 0.2	69.7 \pm 0.3	93.0 \pm 0.1	51.9 \pm 0.7	83.6 \pm 0.3	53.5 \pm 0.1	92.6 \pm 0.2
BEATs-iter3 [58]	AS	90.0	94.0 \pm 0.8	94.7 \pm 0.1	95.8 \pm 0.1	69.4 \pm 0.8	67.3 \pm 0.2	68.0 \pm 0.2	85.2 \pm 0.3	38.5 \pm 1.0	83.7 \pm 0.3	53.6 \pm 0.2	85.9 \pm 0.3
SSAST-Style MSMs													
SSAST [16]	Mix	89.0	93.4 \pm 0.9	96.7 \pm 0.1	96.3 \pm 0.1	66.8 \pm 0.7	56.5 \pm 0.2	60.7 \pm 0.3	53.5 \pm 1.3	28.5 \pm 0.9	68.4 \pm 0.4	38.2 \pm 0.1	71.9 \pm 0.2
SSAST-Tiny	AS	5.4	90.4 \pm 0.7	95.7 \pm 0.1	94.3 \pm 0.1	61.2 \pm 0.5	46.9 \pm 0.2	42.7 \pm 0.2	50.6 \pm 1.6	13.8 \pm 1.0	42.4 \pm 0.6	24.6 \pm 0.1	55.3 \pm 0.2
SSAM-Tiny [20]	AS	4.8	93.7 \pm 0.8	97.1 \pm 0.1	94.9 \pm 0.1	62.0 \pm 0.7	61.8 \pm 0.3	59.2 \pm 0.4	74.8 \pm 0.4	27.8 \pm 1.0	70.6 \pm 0.2	41.3 \pm 0.2	75.0 \pm 0.2
AxLSTM-Tiny [44]	AS	4.3	93.9 \pm 0.7	96.8 \pm 0.1	94.9 \pm 0.1	62.5 \pm 0.6	57.5 \pm 0.2	55.1 \pm 0.5	69.0 \pm 1.6	24.1 \pm 0.6	61.3 \pm 0.6	37.5 \pm 0.2	70.1 \pm 0.2
SSAST-Small	AS	21.5	93.2 \pm 0.5	96.2 \pm 0.1	95.0 \pm 0.1	63.8 \pm 0.4	51.6 \pm 0.2	50.0 \pm 0.3	58.3 \pm 1.2	15.6 \pm 0.7	50.1 \pm 0.6	31.6 \pm 0.1	62.5 \pm 0.2
SSAM-Small [20]	AS	17.9	94.0 \pm 0.7	97.5 \pm 0.1	96.7 \pm 0.1	66.3 \pm 0.8	67.5 \pm 0.2	60.5 \pm 0.3	83.7 \pm 0.3	39.6 \pm 0.7	78.7 \pm 0.6	48.5 \pm 0.1	82.8 \pm 0.3
AxLSTM-Small [44]	AS	16.7	92.9 \pm 1.0	97.4 \pm 0.1	96.6 \pm 0.1	66.6 \pm 0.4	65.0 \pm 0.2	60.3 \pm 0.3	80.5 \pm 0.4	36.5 \pm 0.7	75.5 \pm 0.4	46.5 \pm 0.1	80.4 \pm 0.3
SSAST-Base	AS	85.7	93.1 \pm 0.7	96.6 \pm 0.1	96.2 \pm 0.2	64.6 \pm 0.8	56.0 \pm 0.4	52.9 \pm 0.3	66.1 \pm 1.0	19.2 \pm 0.9	59.6 \pm 0.7	37.5 \pm 0.1	68.2 \pm 0.3
SSAM-Base [20]	AS	69.3	93.2 \pm 1.1	97.7 \pm 0.1	96.9 \pm 0.1	70.5 \pm 0.5	70.3 \pm 0.2	63.5 \pm 0.2	87.9 \pm 0.3	50.4 \pm 0.7	81.0 \pm 0.3	52.2 \pm 0.1	87.9 \pm 0.3
AxLSTM-Base [44]	AS	65.6	93.6 \pm 0.9	97.5 \pm 0.1	97.5 \pm 0.1	71.4 \pm 0.8	68.7 \pm 0.2	63.2 \pm 0.3	85.1 \pm 0.2	43.5 \pm 0.6	79.2 \pm 0.6	51.0 \pm 0.1	85.8 \pm 0.2

overall score $s(m)$, there are more nuances in their performance characteristics on specific types of audio tasks. SSAM models perform extremely well in Speech-based and Audio classification tasks, especially for emotion recognition (CD) and spoken language identification (VL), whereas AxLSTMs fare better for music and pitch perception tasks (BO, Mri-T, NSynth-5h). While neither Mamba-based SSAMs nor xLSTM-based AxLSTMs outperform top of the line MAE-style MSMs, they consistently outperform directly comparable standard Transformer based counterparts, highlighting their viability in learning self-supervised audio representations.

B. Ablations

In this section, we compare Transformer, Mamba and xLSTM based MSMs and see how they fare as we manipulate two key hyperparameters: patch size and input sequence length during pretraining. For more ablations related to selection of other hyperparameters, refer to [19], [44].

Patch size: As mentioned in Section II-A, patch size is a key hyperparameter. Not only does it impact the length of the input, it also impacts the time-frequency resolution of the patches that are processed by the MSM, and thus, impacting model characteristics. The objective of this ablation is not only to see which models maintain performance with different

TABLE III
PATCH SIZE ABLATIONS WITH THE TINY CONFIGURATION

Model	Patch parameters	# Patches	$s(m)$
SSAST-Tiny	$f = 16, t = 8$	125	46.8 \pm 0.3
SSAM-Tiny	$f = 16, t = 8$	125	57.4\pm0.3
AxLSTM-Tiny	$f = 16, t = 8$	125	55.1 \pm 0.2
SSAST-Tiny	$f = 16, t = 4$	250	55.3 \pm 0.2
SSAM-Tiny	$f = 16, t = 4$	250	75.0\pm0.2
AxLSTM-Tiny	$f = 16, t = 4$	250	70.1 \pm 0.2
SSAST-Tiny	$f = 8, t = 4$	500	53.5 \pm 0.3
SSAM-Tiny	$f = 8, t = 4$	500	74.8\pm0.2
AxLSTM-Tiny	$f = 8, t = 4$	500	72.2 \pm 0.3

patch sizes, thus indicating better capability to model distinct time-frequency resolutions, but also to observe how distinct sequence lengths impact performance. To this end, we pretrain and compare SSAST, SSAM and AxLSTM tiny configurations with 3 different patch configurations: (i) $f = 16, t = 8$, (ii), $f = 16, t = 4$ (default) and (iii) $f = 8, t = 4$, where conditions (i) and (iii) represent worsening frequency resolution and improving frequency resolution compared to the default setting (ii), respectively. From Table III, we can see that SSAMs and AxLSTMs outperform standard Transformer

TABLE IV
COMPARING SSAST, SSAM AND AxLSTM OVERALL PERFORMANCE
WITH VARYING AUDIO CLIP DURATIONS

Model	duration (seconds)	# Patches	$s(m)$
SSAST-Tiny	2	250	55.3 \pm 0.2
SSAM-Tiny	2	250	75.0 \pm 0.2
AxLSTM-Tiny	2	250	70.1 \pm 0.2
SSAST-Tiny	5	625	53.4 \pm 0.3
SSAM-Tiny	5	625	75.4 \pm 0.2
AxLSTM-Tiny	5	625	70.3 \pm 0.4
SSAST-Tiny	10	1250	46.4 \pm 0.4
SSAM-Tiny	10	1250	69.6 \pm 0.2
AxLSTM-Tiny	10	1250	OOM

based SSAST across all patch settings. It is worth noting that while decreasing time resolution ($f = 16, t = 8$) worsens performance for all the models, SSAMs observe the biggest performance drop when time resolution is worsened. On the other hand increasing frequency resolution $f = 8, t = 4$ only helps the overall score of the AxLSTM-Tiny model.

Input sequence length during pretraining: In an attempt to tie up the impact of sequence length on downstream performance, we conduct a final ablation experiment where we keep the default patch configuration ($f = 16, t = 4$) while increasing only the duration of audio clip at the time of pretraining. AudioSet has clips of up to 10 seconds in duration, and by default we select a 2-second random crop during pretraining. So we conduct experiments for two additional settings: randomly clipping 5 and 10 second audio clips. From Table IV, we can see that model performance improves for SSAM-Tiny when going from 2-second audio clips to 5-second audio clips, whereas there is very marginal improvement in performance for AxLSTMs. On the other hand, increasing clip duration consistently worsens performance for SSAST-Tiny models. Increasing clip duration any further worsens performance drastically for SSAMs, potentially indicating that we might have hit the performance limit for this model size. AxLSTM-Tiny threw an Out of Memory (OOM) error when attempting to train on 10-second inputs.

OOM issues notwithstanding, results from these two ablations indicate that SSAMs and AxLSTMs scale better with increasing sequence length compared to SSASTs, and are also more parameter efficient.

VI. THINGS WE COULD NOT COVER

In this work, we discuss masked spectrogram modeling and recent advances in neural sequence modeling in context of learning self-supervised general-purpose audio representations. We discuss the two most prominent frameworks for masked spectrogram modeling, viz. SSAST-style and MAE-style MSMs, as well as dwelling into selective state space models (Mamba) and extended long short-term memory models (xLSTMs), which we put to the test against the Transformer architecture in a unified, reproducible MSM framework. However, we did not dive deeper into throughput and efficiency characteristics of the models. Efficiency and throughput is very

hardware dependent, with certain implementations and kernels working better on certain hardware versus others. Testing all available implementations on a multitude of hardware devices in a multitude of sequence length settings is beyond the scope of this paper, whereas testing in a restricted scope have already been done for both Mamba [36], [37], [40] and xLSTM [120]. Furthermore, there are too many nuances to keep track of for effective throughput analysis: things such as the interplay of JIT or runtime compilation, fused implementations, and even running on shared hardware in a university cluster setting, can obfuscate throughput analysis. We thus refrained from conducting a thorough throughput analysis in the paper.

Further, our analysis excludes experiments on full supervised finetuning of the models. While it might have been useful for certain readers and applications, running full finetuning experiments on all these models across all the datasets evaluated was simply infeasible. Besides, we firmly believe that the conducted experiments serve as a solid proxy for the expected level of performance on similar/adjacent tasks, and with our accompanying code and pretrained models, the interested readers would be able to conduct those experiments themselves.

VII. CONCLUSIONS

In this work, we have presented a systematic overview of two key topics: (i) masked spectrogram modeling for learning self-supervised audio representation, and (ii) recent advances in neural sequence modeling architectures, viz. Mamba and xLSTM. The core objective of the paper is to present the reader a means to navigate the research landscape surrounding these topics as well as at the intersection of these topics, while laying down a solid foundation of the key underlying concepts at the heart of these topics. We provide a succinct yet thorough description of masked spectrogram modeling and the two main classes of masked spectrogram models (MSMs). Further, we provide a solid coverage of the fundamentals behind Mamba and xLSTM, the two most promising recent advances in neural sequence processing. We also provide an evaluation of Transformer, Mamba and xLSTMs for learning self-supervised audio representations through masked spectrogram modeling in a unified and reproducible framework, where through empirical analysis of ten varied downstream tasks, we establish the effectiveness of Mamba and xLSTM based approaches over a standard Transformer. All this was accompanied by a continued discussion about potential avenues for future research, and we hope that we inspired new research in the field.

ACKNOWLEDGMENTS

We would like to thank the Pioneer Centre for AI for supporting this research.

REFERENCES

- [1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

- [2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [3] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=H1eA7AEtVS>
- [4] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 8440–8451. [Online]. Available: <https://aclanthology.org/2020.acl-main.747/>
- [5] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, H. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.
- [6] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, "Simmim: A simple framework for masked image modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9653–9663.
- [7] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16000–16009.
- [8] H. Bao, L. Dong, S. Piao, and F. Wei, "BEit: BERT pre-training of image transformers," in *International Conference on Learning Representations*, 2022.
- [9] Z. Tong, Y. Song, J. Wang, and L. Wang, "Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training," *Advances in neural information processing systems*, vol. 35, pp. 10078–10093, 2022.
- [10] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12449–12460.
- [11] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE Transactions on Audio, Speech, and Language Processing*, pp. 1–1, 2021.
- [12] Y.-A. Chung, Y. Zhang, W. Han, C.-C. Chiu, J. Qin, R. Pang, and Y. Wu, "w2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 244–250.
- [13] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1505–1518, 2022.
- [14] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "data2vec: A general framework for self-supervised learning in speech, vision and language," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 1298–1312.
- [15] A. Baevski, A. Babu, W.-N. Hsu, and M. Auli, "Efficient self-supervised learning with contextualized target representations for vision, speech and language," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23. JMLR.org, 2023.
- [16] Y. Gong, C.-I. Lai, Y.-A. Chung, and J. Glass, "Ssast: Self-supervised audio spectrogram transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 10699–10709.
- [17] P.-Y. Huang, H. Xu, J. Li, A. Baevski, M. Auli, W. Galuba, F. Metzger, and C. Feichtenhofer, "Masked autoencoders that listen," *Advances in Neural Information Processing Systems*, vol. 35, pp. 28708–28720, 2022.
- [18] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Masked spectrogram modeling using masked autoencoders for learning general-purpose audio representation," in *HEAR: Holistic Evaluation of Audio Representations*. PMLR, 2022, pp. 1–24.
- [19] S. Yadav, S. Theodoridis, L. K. Hansen, and Z.-H. Tan, "Masked autoencoders with multi-window local-global attention are better audio learners," in *The Twelfth International Conference on Learning Representations*, 2024.
- [20] S. Yadav and Z.-H. Tan, "Audio mamba: Selective state spaces for self-supervised audio representations," in *Interspeech 2024*, 2024, pp. 552–556.
- [21] A. Baade, P. Peng, and D. Harwath, "MAE-AST: Masked Autoencoding Audio Spectrogram Transformer," in *Interspeech 2022*. ISCA, Sep. 2022, pp. 2438–2442.
- [22] H. Dinkel, Z. Yan, Y. Wang, J. Zhang, Y. Wang, and B. Wang, "Scaling up masked audio encoder learning for general audio classification," in *Interspeech 2024*, 2024, pp. 547–551.
- [23] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International conference on machine learning*. PMLR, 2020, pp. 5156–5165.
- [24] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.
- [25] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Re, "Flashattention: Fast and memory-efficient exact attention with IO-awareness," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.
- [26] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," in *The Twelfth International Conference on Learning Representations*, 2024.
- [27] S. Alberti, N. Dorn, L. Thesing, and G. Kutyniok, "Sumformer: Universal approximation for efficient transformers," in *Topological, Algebraic and Geometric Learning Workshops 2023*. PMLR, 2023, pp. 72–86.
- [28] F. Mai, A. Pannatier, F. Fehr, H. Chen, F. Marelli, F. Fleuret, and J. Henderson, "Hypermixer: An mlp-based low cost alternative to transformers," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 15632–15654.
- [29] A. Gu, K. Goel, and C. Re, "Efficiently modeling long sequences with structured state spaces," in *International Conference on Learning Representations*, 2022.
- [30] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. K. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "xLSTM: Extended long short-term memory," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [31] E. Nguyen, K. Goel, A. Gu, G. Downs, P. Shah, T. Dao, S. Baccus, and C. Ré, "S4ND: Modeling images and videos as multidimensional signals with state spaces," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.
- [32] 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 *The Eleventh International Conference on Learning Representations*, 2023.
- [33] M. Poli, S. Massaroli, E. Nguyen, D. Y. Fu, T. Dao, S. Baccus, Y. Bengio, S. Ermon, and C. Ré, "Hyena hierarchy: Towards larger convolutional language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 28043–28078.
- [34] Y. Sun, L. Dong, S. Huang, S. Ma, Y. Xia, J. Xue, J. Wang, and F. Wei, "Retentive network: A successor to transformer for large language models," *arXiv preprint arXiv:2307.08621*, 2023.
- [35] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, L. Derczynski *et al.*, "Rwkv: Reinventing rnns for the transformer era," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 14048–14077.
- [36] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.
- [37] T. Dao and A. Gu, "Transformers are ssms: generalized models and efficient algorithms through structured state space duality," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.
- [38] X. Gao and N. F. Chen, "Speech-mamba: Long-context speech recognition with selective state spaces models," in *2024 IEEE Spoken Language Technology Workshop (SLT)*, 2024, pp. 1–8.
- [39] K. Li, G. Chen, R. Yang, and X. Hu, "Spmamba: State-space model is all you need in speech separation," *arXiv preprint arXiv:2404.02063*, 2024.
- [40] S. Shams, S. S. Dindar, X. Jiang, and N. Mesgarani, "Ssamba: Self-supervised audio representation learning with mamba state space model," in *2024 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2024, pp. 1053–1059.

- [41] M. H. Erol, A. Senocak, J. Feng, and J. S. Chung, "Audio mamba: Bidirectional state space model for audio representation learning," *IEEE Signal Processing Letters*, vol. 31, pp. 2975–2979, 2024.
- [42] D. Mu, Z. Zhang, H. Yue, Z. Wang, J. Tang, and J. Yin, "Seld-mamba: Selective state-space model for sound event localization and detection with source distance estimation," *arXiv preprint arXiv:2408.05057*, 2024.
- [43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [44] Sarthak Yadav and Sergios Theodoridis and Zheng-Hua Tan, "AxL-STMs: learning self-supervised audio representations with xLSTMs," in *Interspeech 2025*, 2025, pp. 3524–3528.
- [45] N. L. Kühne, J. Østergaard, J. Jensen, and Z.-H. Tan, "xlstm-senet: xlstm for single-channel speech enhancement," *arXiv preprint arXiv:2501.06146*, 2025.
- [46] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski, "Vision transformers need registers," in *The Twelfth International Conference on Learning Representations*, 2024.
- [47] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.
- [48] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6419–6423.
- [49] A. T. Liu, S.-W. Li, and H.-y. Lee, "Tera: Self-supervised learning of transformer encoder representation for speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2351–2366, 2021.
- [50] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword transformer: A self-attention model for keyword spotting," in *Interspeech 2021*. ISCA, 2021, pp. 4249–4253.
- [51] H. S. Bovbjerg and Z.-H. Tan, "Improving label-deficient keyword spotting through self-supervised pretraining," in *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, 2023, pp. 1–5.
- [52] J. Mørk, H. S. Bovbjerg, G. Kiss, and Z.-H. Tan, "Noise-robust keyword spotting through self-supervised pretraining," 2024.
- [53] G. Li, H. Zheng, D. Liu, C. Wang, B. Su, and C. Zheng, "Sem-MAE: Semantic-guided masking for learning masked autoencoders," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.
- [54] J. Shin, I. Lee, J. Lee, and J. Lee, "Self-guided masked autoencoder," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [55] 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 *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 9992–10002.
- [56] P. Gao, T. Ma, H. Li, Z. Lin, J. Dai, and Y. Qiao, "Mcmae: masked convolution meets masked autoencoders," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.
- [57] S. Deshmukh, B. Elizalde, R. Singh, and H. Wang, "Pengi: An audio language model for audio tasks," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [58] S. Chen, Y. Wu, C. Wang, S. Liu, D. Tompkins, Z. Chen, W. Che, X. Yu, and F. Wei, "Beats: audio pre-training with acoustic tokenizers," in *Proceedings of the 40th International Conference on Machine Learning*, 2023, pp. 5178–5193.
- [59] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Interspeech 2020*, 2020, pp. 5036–5040.
- [60] K. M. Choromanski, V. Likhoshesterov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller, "Rethinking attention with performers," in *International Conference on Learning Representations*, 2021.
- [61] W. Meng, Y. Luo, X. Li, D. Jiang, and Z. Zhang, "Polaformer: Polarity-aware linear attention for vision transformers," in *The Thirteenth International Conference on Learning Representations*, 2025.
- [62] N. Shazeer, "Fast transformer decoding: One write-head is all you need," 2019.
- [63] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebron, and S. Sanghai, "GQA: Training generalized multi-query transformer models from multi-head checkpoints," in *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [64] H. Liu, M. Zaharia, and P. Abbeel, "Ringattention with blockwise transformers for near-infinite context," in *The Twelfth International Conference on Learning Representations*, 2024.
- [65] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "FNet: Mixing tokens with Fourier transforms," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 4296–4313.
- [66] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent memory with optimal polynomial projections," *Advances in neural information processing systems*, vol. 33, pp. 1474–1487, 2020.
- [67] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long range arena: A benchmark for efficient transformers," in *International Conference on Learning Representations*, 2021.
- [68] Y. Liu, Y. Tian, Y. Zhao, H. Yu, L. Xie, Y. Wang, Q. Ye, J. Jiao, and Y. Liu, "Vmamba: Visual state space model," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 103031–103063.
- [69] 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 *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.
- [70] W. Yu and X. Wang, "Mambaout: Do we really need mamba for vision?" in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 4484–4496.
- [71] K. Li, X. Li, Y. Wang, Y. He, Y. Wang, L. Wang, and Y. Qiao, "Videomamba: State space model for efficient video understanding," in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds. Cham: Springer Nature Switzerland, 2025, pp. 237–255.
- [72] D. Liang, X. Zhou, W. Xu, X. Zhu, Z. Zou, X. Ye, X. Tan, and X. Bai, "Pointmamba: A simple state space model for point cloud analysis," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 32653–32677.
- [73] T. Zhang, H. Yuan, L. Qi, J. Zhang, Q. Zhou, S. Ji, S. Yan, and X. Li, "Point cloud mamba: Point cloud learning via state space model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 10, 2025, pp. 10121–10130.
- [74] Y. Schiff, C.-H. Kao, A. Gokaslan, T. Dao, A. Gu, and V. Kuleshov, "Caduceus: Bi-directional equivariant long-range dna sequence modeling," *Proceedings of machine learning research*, vol. 235, p. 43632, 2024.
- [75] J. Ma, F. Li, and B. Wang, "U-mamba: Enhancing long-range dependency for biomedical image segmentation," *arXiv preprint arXiv:2401.04722*, 2024.
- [76] T. Dao and A. Gu, "Transformers are ssms: generalized models and efficient algorithms through structured state space duality," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.
- [77] J. Wang, D. Paliotta, A. May, A. M. Rush, and T. Dao, "The mamba in the llama: Distilling and accelerating hybrid models," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 62432–62457.
- [78] B. Lenz, O. Lieber, A. Arazi, A. Bergman, A. Manevich, B. Peleg, B. Aviram, C. Almagor, C. Fridman, D. Padnos, D. Gissin, D. Jannai, D. Muhlga, D. Zimberg, E. M. Gerber, E. Dolev, E. Krakovsky, E. Safahi, E. Schwartz, G. Cohen, G. Shachaf, H. Rozenblum, H. Bata, I. Blass, I. Magar, I. Dalmedigos, J. Osin, J. Fadlon, M. Rozman, M. Danos, M. Gokhman, M. Zusman, N. Gidron, N. Ratner, N. Gat, N. Rozen, O. Fried, O. Leshno, O. Antverg, O. Abend, O. Dagan, O. Cohavi, R. Alon, R. Belson, R. Cohen, R. Gilad, R. Glozman, S. Lev, S. Shalev-Shwartz, S. H. Meirom, T. Delbari, T. Ness, T. Asida, T. B. Gal, T. Braude, U. Pumerantz, J. Cohen, Y. Belinkov, Y. Globerson, Y. P. Levy, and Y. Shoham, "Jamba: Hybrid transformer-mamba language models," in *The Thirteenth International Conference on Learning Representations*, 2025.
- [79] Y. Liu and L. Yi, "Map: Unleashing hybrid mamba-transformer vision backbone's potential with masked autoregressive pretraining," in *Pro-*

- ceedings of the Computer Vision and Pattern Recognition Conference, 2025, pp. 9676–9685.
- [80] S. Ren, X. Li, H. Tu, F. Wang, F. Shu, L. Zhang, J. Mei, L. Yang, P. Wang, H. Wang, A. Yuille, and C. Xie, “Autoregressive pretraining with mamba in vision,” in *The Thirteenth International Conference on Learning Representations*, 2025.
 - [81] S. Hwang, A. Lahoti, R. Puduppully, T. Dao, and A. Gu, “Hydra: Bidirectional state space models through generalized matrix mixers,” in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
 - [82] T. Huang, X. Pei, S. You, F. Wang, C. Qian, and C. Xu, “Localmamba: Visual state space model with windowed selective scan,” in *Computer Vision – ECCV 2024 Workshops*, A. Del Bue, C. Canton, J. Pont-Tuset, and T. Tommasi, Eds. Cham: Springer Nature Switzerland, 2025, pp. 12–22.
 - [83] Y. Masuyama, K. Miyazaki, and M. Murata, “Mamba-based decoder-only approach with bidirectional speech modeling for speech recognition,” in *2024 IEEE Spoken Language Technology Workshop (SLT)*, 2024, pp. 1–6.
 - [84] X. Zhang, Q. Zhang, H. Liu, T. Xiao, X. Qian, B. Ahmed, E. Ambikairajah, H. Li, and J. Epps, “Mamba in speech: Towards an alternative to self-attention,” *IEEE Transactions on Audio, Speech and Language Processing*, 2025.
 - [85] Y. Fang and X. Li, “Mamba for streaming asr combined with unimodal aggregation,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [86] X. Jiang, Y. A. Li, A. Nicolas Florea, C. Han, and N. Mesgarani, “Speech slytherin: Examining the performance and efficiency of mamba for speech separation, recognition, and synthesis,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [87] X. Zhang, J. Ma, M. Shahin, B. Ahmed, and J. Epps, “Rethinking mamba in speech processing by self-supervised models,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [88] T. Luo, F. Zhou, and Z. Bai, “Mambagan: Mamba based metric gan for monaural speech enhancement,” in *2024 International Conference on Asian Language Processing (IALP)*. IEEE, 2024, pp. 411–416.
 - [89] R. Chao, W.-H. Cheng, M. L. Quatra, S. M. Siniscalchi, C.-H. H. Yang, S.-W. Fu, and Y. Tsao, “An investigation of incorporating mamba for speech enhancement,” in *2024 IEEE Spoken Language Technology Workshop (SLT)*, 2024, pp. 302–308.
 - [90] J. Wang, Z. Lin, T. Wang, M. Ge, L. Wang, and J. Dang, “Mamba-seunet: Mamba unet for monaural speech enhancement,” in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.
 - [91] N. L. Kühne, J. Jensen, J. Østergaard, and Z.-H. Tan, “Mambattention: Mamba with multi-head attention for generalizable single-channel speech enhancement,” *arXiv preprint arXiv:2507.00966*, 2025.
 - [92] R. Chao, R. Nasretidov, Y.-C. F. Wang, A. Jukić, S.-W. Fu, and Y. Tsao, “Universal speech enhancement with regression and generative mamba,” *arXiv preprint arXiv:2505.21198*, 2025.
 - [93] X. Qian, J. Gao, Y. Zhang, Q. Zhang, H. Liu, L. P. Garcia, and H. Li, “Sav-se: Scene-aware audio-visual speech enhancement with selective state space model,” *IEEE Journal of Selected Topics in Signal Processing*, 2025.
 - [94] C. Fan, Y. Gao, Z. Pan, J. Zhang, H. Zhang, J. Zhang, and Z. Lv, “Improved feature extraction network for neuro-oriented target speaker extraction,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [95] T. H. Avenstrup, B. Elek, I. L. Mádi, A. B. Schin, M. Mørup, B. S. Jensen, and K. Olsen, “Sepmamba: State-space models for speaker separation using mamba,” in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.
 - [96] A. Plaquet, N. Tawara, M. Delcroix, S. Horiguchi, A. Ando, and S. Araki, “Mamba-based segmentation model for speaker diarization,” in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.
 - [97] Z. Lin, Y. Wang, Y. Zhou, F. Du, and Y. Yang, “Ste-mamba: Automated multimodal depression detection through emotional analysis and spatio-temporal information ensemble,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [98] J. Ye, J. Zhang, and H. Shan, “Depmamba: Progressive fusion mamba for multimodal depression detection,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [99] J. Liu, Y. Shang, M. Yang, Z. Shao, J. Lu, and T. Liu, “Mfmamba: A multimodal fusion state space model for depression recognition,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [100] Y. Lee and C. Kim, “Wave-u-mamba: An end-to-end framework for high-quality and efficient speech super resolution,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [101] C. Zhu, C. Sun, Y. Liu, J. Chen, and K. Luo, “Enhancing speech emotion recognition with speech dynamic modeling and multi-modal knowledge distillation,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [102] W.-J. Lee, F.-C. Hsieh, X. Chen, F.-D. Tsai, and Y.-H. Yang, “Exploring state-space-model based language model in music generation,” *arXiv preprint arXiv:2507.06674*, 2025.
 - [103] J. Chen, X. Tang, T. Xie, J. Wang, W. Dong, and B. Shi, “Musicmamba: A dual-feature modeling approach for generating chinese traditional music with modal precision,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [104] M. F. Colombo, F. Ronchini, L. Comanducci, and F. Antonacci, “Mambafoley: Foley sound generation using selective state-space models,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [105] Y. Chen, J. Yi, J. Xue, C. Wang, X. Zhang, S. Dong, S. Zeng, J. Tao, Z. Lv, and C. Fan, “Rawbmamba: End-to-end bidirectional state space model for audio deepfake detection,” in *Interspeech 2024*, 2024, pp. 2720–2724.
 - [106] Y. E. Kheir, T. Polzehl, and S. Möller, “Bicrossmamba-st: Speech deepfake detection with bidirectional mamba spectro-temporal cross-attention,” *arXiv preprint arXiv:2505.13930*, 2025.
 - [107] J. Liang, B. Meyer, I. N. Lee, and T.-T. Do, “Self-supervised learning for acoustic few-shot classification,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.
 - [108] J. Lin and H. Hu, “Audio mamba: Pretrained audio state space model for audio tagging,” *arXiv preprint arXiv:2405.13636*, 2024.
 - [109] G. Yuksel, M. van Gerven, and K. van der Heijden, “General-purpose audio representation learning for real-world sound scenes,” *arXiv preprint arXiv:2506.00934*, 2025.
 - [110] S. Li, H. Singh, and A. Grover, “Mamba-nd: Selective state space modeling for multi-dimensional data,” in *European Conference on Computer Vision*. Springer, 2024, pp. 75–92.
 - [111] L. Ren, Y. Liu, Y. Lu, Yelong shen, C. Liang, and W. Chen, “Samba: Simple hybrid state space models for efficient unlimited context language modeling,” in *The Thirteenth International Conference on Learning Representations*, 2025.
 - [112] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Advances in neural information processing systems*, vol. 27, 2014.
 - [113] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.
 - [114] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.
 - [115] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, “Deep speech 2: End-to-end speech recognition in english and mandarin,” in *International conference on machine learning*. PMLR, 2016, pp. 173–182.
 - [116] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
 - [117] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *International conference on machine learning*. PMLR, 2015, pp. 2048–2057.
 - [118] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, “Draw: A recurrent neural network for image generation,” in *International conference on machine learning*. PMLR, 2015, pp. 1462–1471.

- [119] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2410–2419.
- [120] M. Beck, K. Pöppel, P. Lippe, R. Kurl, P. M. Blies, G. Klambauer, S. Böck, and S. Hochreiter, "xLSTM 7b: A recurrent LLM for fast and efficient inference," in *Forty-second International Conference on Machine Learning*, 2025. [Online]. Available: <https://openreview.net/forum?id=LV3DpKD08B>
- [121] B. Alkin, M. Beck, K. Pöppel, S. Hochreiter, and J. Brandstetter, "Vision-LSTM: xLSTM as generic vision backbone," in *The Thirteenth International Conference on Learning Representations*, 2025.
- [122] N. Schmidinger, L. Schneckenreiter, P. Seidl, J. Schimunek, P.-J. Hoedt, J. Brandstetter, A. Mayr, S. Luukkonen, S. Hochreiter, and G. Klambauer, "Bio-xLSTM: Generative modeling, representation and in-context learning of biological and chemical sequences," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: <https://openreview.net/forum?id=ljbXZdugdj>
- [123] Q. Zhu, Y. Cai, and L. Fan, "Seg-lstm: performance of xlstm for semantic segmentation of remotely sensed images," *arXiv preprint arXiv:2406.14086*, 2024.
- [124] M. Kraus, F. Divo, D. S. Dhami, and K. Kersting, "xlstm-mixer: Multivariate time series forecasting by mixing via scalar memories," *arXiv preprint arXiv:2410.16928*, 2024.
- [125] Y. Kong, Z. Wang, Y. Nie, T. Zhou, S. Zohren, Y. Liang, P. Sun, and Q. Wen, "Unlocking the power of lstm for long term time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 11, 2025, pp. 11 968–11 976.
- [126] A. Auer, P. Podest, D. Klotz, S. Böck, G. Klambauer, and S. Hochreiter, "Tirex: Zero-shot forecasting across long and short horizons," in *1st ICM Workshop on Foundation Models for Structured Data*, 2025. [Online]. Available: <https://openreview.net/forum?id=cWrbpOZGRa>
- [127] M. Heidari, E. K. Aghdam, A. Manzella, D. Hsu, R. Scalabrino, W. Chen, D. J. Foran, and I. Hacıhaliloğlu, "A study on the performance of u-net modifications in retroperitoneal tumor segmentation," in *Medical Imaging 2025: Computer-Aided Diagnosis*, vol. 13407. SPIE, 2025, pp. 382–391.
- [128] S. Zhu, Y. Chen, S. Jiang, W. Chen, C. Liu, Y. Wang, X. Chen, Y. Ke, F. Qin, C. Wang *et al.*, "Xlstm-hved: Cross-modal brain tumor segmentation and mri reconstruction method using vision xlstm and heteromodal variational encoder-decoder," in *2025 IEEE 22nd International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2025, pp. 1–5.
- [129] T. Chen, C. Ding, L. Zhu, T. Xu, D. Ji, Y. Wang, Y. Zang, and Z. Li, "xlstm-unet can be an effective 2d & 3d medical image segmentation backbone with vision-lstm (vil) better than its mamba counterpart," *arXiv preprint arXiv:2407.01530*, 2024.
- [130] P. Dutta, S. Bose, S. K. Roy, and S. Mitra, "Are vision xlstm embedded unet more reliable in medical 3d image segmentation?" *arXiv preprint arXiv:2406.16993*, 2024.
- [131] M. Beck, K. Pöppel, P. Lippe, and S. Hochreiter, "Tiled flash linear attention: More efficient linear RNN and xLSTM kernels," in *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025. [Online]. Available: <https://openreview.net/forum?id=P6CPFdexbk>
- [132] 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 *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 776–780.
- [133] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.
- [134] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3875–3879.
- [135] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Self-supervised learning for general-purpose audio representation," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.
- [136] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.
- [137] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*. ACM Press, 2015.
- [138] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "Fsd50k: an open dataset of human-labeled sound events," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.
- [139] H. Cao, D. G. Cooper, M. K. Keutmann, R. C. Gur, A. Nenkov, and R. Verma, "Crema-d: Crowd-sourced emotional multimodal actors dataset," *IEEE transactions on affective computing*, vol. 5, no. 4, pp. 377–390, 2014.
- [140] 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.
- [141] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. Jeff Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K.-t. Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H.-y. Lee, "SUPERB: Speech Processing Universal PERFORMANCE Benchmark," in *Proc. Interspeech 2021*, 2021, pp. 1194–1198.
- [142] J. Turian, J. Shier, H. R. Khan, B. Raj, B. W. Schuller, C. J. Steinmetz, C. Malloy, G. Tzanetakis, G. Velarde, K. McNally, M. Henry, N. Pinto, C. Noufi, C. Clough, D. Herremans, E. Fonseca, J. Engel, J. Salamon, P. Esling, P. Manocha, S. Watanabe, Z. Jin, and Y. Bisk, "HEAR: Holistic Evaluation of Audio Representations," in *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*. PMLR, Jul. 2022, pp. 125–145, iSSN: 2640-3498.
- [143] M. Tian, A. Srinivasamurthy, M. Sandler, and X. Serra, "A study of instrument-wise onset detection in beijing opera percussion ensembles," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 2159–2163.
- [144] A. Anantapadmanabhan, A. Bellur, and H. A. Murthy, "Modal analysis and transcription of strokes of the mridangam using non-negative matrix factorization," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.
- [145] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," 2017.
- [146] F.-R. Stöter, S. Chakrabarty, B. Edler, and E. A. Habets, "Classification vs. regression in supervised learning for single channel speaker count estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.
- [147] B. Kim, M. Ghei, B. Pardo, and Z. Duan, "Vocal imitation set: a dataset of vocally imitated sound events using the audioset ontology," in *DCASE*, 2018, pp. 148–152.
- [148] 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," in *International Conference on Learning Representations*, 2021.
- [149] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [150] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=Bkg6RiCqY7>
- [151] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient Training of Audio Transformers with Patchout," in *Proc. Interspeech 2022*, 2022, pp. 2753–2757.
- [152] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," in *Interspeech 2021*, 2021, pp. 571–575.