MoMamba: A Lightweight Music Oriented Mamba-Based Model for Music Information Retrieval Tasks

Anonymous authors
Paper under double-blind review

Abstract

Music Information Retrieval (MIR) tasks on raw audio have traditionally been tackled using convolutional neural networks (CNNs) and transformer-based models. While CNNs effectively capture local structures and transformers leverage attention for long-range dependencies, both architectures come with computational and scalability challenges. Recently, State-Space models, such as Mamba, have become popular for long time series data such as audio, and have shown considerable promise compared to convolutional or transformer-based architectures. In this study, we introduce a novel extension of Mamba tailored to music. Our resulting method, MoMamba (Music Oriented Mamba), is a lightweight Mamba-based music classification model. We evaluate MoMamba's performance across several benchmark MIR tasks. Our results show that MoMamba consistently outperforms a number of baselines, including an existing Mamba-based method, on all of the benchmark datasets we considered. Importantly, all models were trained from scratch without any pretraining, making the performance gains especially notable since they cannot be attributed to transfer learning. Additionally, our model's performance rivals existing benchmarks from models pretrained on much larger datasets. Our work highlights the advantages of MoMamba in music analysis and retrieval such as accuracy and inference time, encouraging further research into its capabilities within the MIR domain.

1 Introduction

Extracting features from music is essential for understanding its structure, patterns, and meaning. These features enable various applications, from classification and recommendation systems to composition and analysis (Copet et al., 2023; Dhariwal et al., 2020; Schedl et al., 2018; Hernandez-Olivan & Beltran, 2021). By identifying musical characteristics, we can enhance both human and machine interpretation, thereby improving organization, retrieval, and interaction with music.

Traditionally, deep learning approaches for music downstream classification tasks have relied heavily on Convolutional Neural Networks (CNNs) (Elbir & Aydin, 2020; Bian et al., 2019). CNNs are particularly effective in processing spectrogram representations of audio but struggle to capture global context effectively. Convolutional Recurrent Neural Networks (CRNNs) (Choi et al., 2017) combine CNNs for feature extraction with Recurrent Neural Networks (RNNs) for sequential modeling. While CRNNs have demonstrated strong performance in MIR tasks (Choi et al., 2017), the design of the model inherently has challenges such as vanishing/exploding gradients and global locality problems. More recently, attention-based models have emerged, leading to a shift toward Audio Spectrogram Transformers (ASTs) (Gong et al., 2021). They leverage self-attention mechanisms to model long-range dependencies and capture global context. However, they come with high computational cost, require substantial pretraining data, and have quadratic complexity.

Recently, State-Space Models (SSMs) have gained attention in natural language processing (NLP) due to their ability to model long-range dependencies while maintaining computational efficiency (Gu & Dao, 2024). Their success in NLP has sparked interest in their applicability to other sequential domains, including audio and music processing. While the use of SSMs in audio is still in its early stages, recent efforts have demonstrated their potential in various tasks. Notable work such as AudioMamba (Erol et al., 2024) has

showcased promising results in classifying speech and sound effects, illustrating SSMs' ability to effectively model temporal structures in audio signals. These advancements suggest that SSMs provide a viable alternative to traditional deep learning architectures, particularly for music-related applications where capturing both local and long-range dependencies is essential.

A key limitation of existing Mamba-based approaches is that they retain Transformer-based ideas, such as patchification and positional encoding, simply replacing attention blocks with Mamba rather than leveraging Mamba's native sequential processing. To address this, our method utilizes the entire spectrogram without patchification in order to keep the spectrogram as one intact sequence and remove positional encoding. When comparing downstream task performance to AudioMamba (Erol et al., 2024), we can see that the metric for every task is higher. We also see that when comparing to AST (Gong et al., 2021), the average metric across tasks for our method is higher and the inference speed is significantly faster.

We note that pretrained foundational models have demonstrated strong performance on the MIR tasks we consider (Li et al., 2024; Castellon et al., 2021; McCallum et al., 2022). However, these methods rely on large-scale external datasets, which makes it difficult to separate the effects of the architecture from the benefits of pretraining. For this reason, our main comparisons are with non-pretrained models, but we also provide the comparison to MERT (Li et al., 2024) and pretrained models from (McCallum et al., 2022). Once large pretrained Mamba models become available, a more direct comparison with other pretrained approaches will be possible.

We evaluate multiple model setups to determine the effects of each component. Based on these experiments, our contributions include developing MoMamba, a Mamba-based music classification model with a class token, and demonstrating its strong performance across multiple downstream music tasks.

Our contributions:

- **We develop a Mamba-based music classification model with a class token mechanism (MoMamba):** We leverage the Mamba architecture to process **entire** audio spectrograms, and introduce a class token at the **end** of the sequence to enhance classification performance. To our knowledge, we are the first to apply Mamba to these MIR downstream tasks.

- **We demonstrate that MoMamba achieves competitive performance on downstream tasks:** Through extensive experiments with training from scratch, MoMamba matches or surpasses the performance of existing models on seven music classification benchmarks, while also offering superior inference time when compared to ASTs and RNNs. MoMamba also outperforms AudioMamba in every metric, and performs comparably to pretrained foundational models that utilize orders of magnitude more training data.

2 Related Work

Early deep learning approaches to raw audio music classification are primarily CNN-based, leveraging architectures such as ResNet18 (Elbir & Aydin, 2020) and DenseNet (Bian et al., 2019). These models typically process spectrograms that have been compressed into smaller images (e.g., 128×128) before being fed into the network. In meter detection, ResNet18 demonstrates superior performance to both traditional non-machine-learning techniques and supervised non-deep-learning models (Abimbola et al., 2024).

To address the sequential nature of music data, CRNNs serve as a hybrid approach (Choi et al., 2017), combining CNNs for feature extraction with RNNs for temporal modeling. This architecture allows for more effective processing of spectrograms by capturing both spatial and sequential patterns, making it particularly well-suited for MIR downstream tasks. By leveraging RNN components such as LSTMs or GRUs, CRNNs improve upon pure CNN models by incorporating temporal dependencies.

ASTs have emerged as a more effective alternative for audio-related tasks thanks to their attention mechanism. Unlike CNNs, which rely on localized feature extraction, transformers process entire sequences, enabling them to capture long-range dependencies and improve overall performance (Gong et al., 2021).

Transformer-based models have driven major advances in music generation and analysis, with notable impact on downstream MIR tasks. For example, MusicGen (Copet et al., 2023) is an autoregressive transformer that uses EnCodec (Défossez et al., 2022) for audio tokenization, and its internal representations have been shown to transfer effectively to tasks such as instrument recognition and music tagging (Koo et al., 2024). JukeBox (Dhariwal et al., 2020) takes a different approach, employing a sparse transformer over multi-scale VQ-VAE codes to generate raw audio with singing and achieving coherence over several minutes. Beyond generation, the learned representations from JukeBox have also proven useful for MIR tasks like genre classification and emotion recognition (Castellon et al., 2021). Together, these works illustrate the versatility of transformer-based models in capturing rich musical structure and providing transferable features for analysis.

However, training such models requires extensive datasets. For instance, MusicGen was trained on 20,000 hours of licensed music (Copet et al., 2023), and JukeBox utilizes a dataset of 1.2 million songs (Dhariwal et al., 2020). This poses challenges for MIR tasks with limited labeled data, as small datasets can lead to overfitting and hinder the models' ability to generalize effectively. To mitigate this, techniques such as data augmentation, transfer learning, and the use of synthetic data have been explored.

More recently, pretrained models such as MERT (Li et al., 2024) and MuQ (Zhu et al., 2025) have been introduced as foundation models for music understanding. Both MERT and MuQ are trained on 160 thousand hours of music audio and leverages self-supervised pretraining to provide general-purpose embeddings that have proven highly effective across a range of MIR benchmarks. These models represent an exciting direction, but their reliance on massive pretraining corpora raises questions about accessibility and applicability in domains where such resources are unavailable.

AudioMamba (Erol et al., 2024), a Mamba-based spectrogram classifier, performs well in speech and sound effects (SFX) classification tasks. Furthermore, AudioMamba shows that Mamba-based models are more memory efficient than transformer-based models. This success suggests that Mamba-based architectures could offer advantages in sequential audio modeling. However, unlike speech or short sound effects, music often contains long-range harmonic and rhythmic structures. Since the general design of AudioMamba is very similar to ASTs, it ignores a main benefit of Mamba-based models: the ability to process long spectrograms sequentially. We address this problem with the design of MoMamba. In this study, we explore Mamba-based architectures for MIR tasks, evaluating their effectiveness across classification benchmarks to assess their viability against other architectures.

3 Model Approach

State Space Models (SSMs) (Gu et al., 2021) are mathematical models designed for sequence-to-sequence transformations. In general, SSMs, including Mamba, map an input signal $u \in \mathbb{R}^D$ to a hidden state $h \in \mathbb{R}^H$, and then to an output signal $y \in \mathbb{R}^D$, where H is the hidden state size and D is the number of channels.

3.1 Preliminaries

There are four continuous parameters used in the Mamba model (Gu & Dao, 2024): $\Delta \in \mathbb{R}, A \in \mathbb{R}^{H\times H}, B \in \mathbb{R}^{H\times D}, C \in \mathbb{R}^{D\times H}$, where we use the frequency bins as channels. Δ controls the discretization resolution, A is the state transition matrix, B maps the input sequence to the hidden state, and C maps the hidden state to the output state. Specifically, Δ changes the focus on the current timestep u_t: larger values emphasize u_t, while smaller values allow the previous state to dominate. In Mamba, the parameters $\Delta, B,$ and C are dependent on the input signal. To apply them effectively, the model must first be discretized, converting its continuous-time parameters into discrete counterparts by using fixed formulas $f_a(\Delta, A)$ and $f_b(\Delta, A, B)$. Mamba's discretization (Gu & Dao, 2024) is formulated as:

$$\begin{aligned} \overline{A} &= f_a(\Delta, A) = \exp(\Delta A) \\ \overline{B} &= f_b(\Delta, A, B) = (\Delta A)^{-1}(\overline{A} - I) \cdot (\Delta B). \end{aligned} \tag{1}$$

C does not need to be discretized since it is just a linear projection from the hidden state to the output. Finally, we have four learnable parameters: $\Delta, \overline{A}, \overline{B}, C$.

Once discretized, these parameters allow for efficient sequence modeling. Let $u^{(d)} \in \mathbb{R}^L$ be a sequence for the d-th frequency bin, where L represents the number of timesteps in the spectrogram. The sequence transformation can be efficiently computed with the kernel $\overline{K}$ defined below. Using the discretized parameters, the SSM updates its hidden state and computes the output as follows:

$$\begin{aligned} h_t^{(d)} &= \overline{A} h_{t-1}^{(d)} + \overline{B} u_t^{(d)} \\ y_t^{(d)} &= C h_t^{(d)} \\ \overline{K} &= (C\overline{B}, C\overline{AB}, \dots, C\overline{A}^k\overline{B}, \dots) \\ y^{(d)} &= \overline{K} \cdot u^{(d)}. \end{aligned} \tag{2}$$

The output $y^{(d)} \in \mathbb{R}^L$ learns information throughout the whole sequence. For more details, refer to (Gu & Dao, 2024).

3.2 MoMamba Architecture

MoMamba first projects the spectrogram generated from the audio sample into a higher-dimensional space, then appends a class token. The resulting features pass through N bidirectional Mamba layers. Afterward, the class token is extracted by selecting the final timestep of the latent representation. Finally, the resulting vector is projected to the class dimension, and applying softmax produces a probability distribution.

One of the main differences between MoMamba and AudioMamba is the length of each sample. Since AudioMamba is evaluated on speech and sound effects, samples in their work are 10 seconds long, resulting in a shorter spectrogram (Erol et al., 2024). However, the spectrograms used for MIR tasks are often longer, around 30 seconds or longer (Abimbola et al., 2023; Tzanetakis & Cook, 2002; Knees et al., 2015). In addition, MoMamba removes the use of positional encoding, due to the fact that Mamba processes the spectrogram sequentially across the temporal dimension. Furthermore, MoMamba does not patchify the input spectrogram, removing another reason to utilize positional encoding. By patchifying inputs and adding unnecessary positional encoding, AudioMamba fails to exploit the strengths of SSMs. We provide ablations with these claims in Table 1. With these changes to the overall structure, we show that MoMamba outperforms AudioMamba in MIR tasks.

3.2.1 Classification Token

Rather than patching the spectrogram as in AudioMamba (Erol et al., 2024), we keep the whole sequence in order to preserve temporal continuity. The features are embedded into a higher-dimensional space. Then, a class token is appended to the end of the sequence. The class token is a parameter that does not contain the label of the sample, rather, it is the token that accumulates the information from the entire sequence. Classification tokens are regularly used in classification models such as BERT (Devlin et al., 2019) and AudioMamba (Erol et al., 2024), where the information is aggregated into the token.

Let $c \in \mathbb{R}^D$ be the classification tokens. We append this to the input signal $u \in \mathbb{R}^{L \times D}$ to form $u' = (u_0, u_1, \dots, u_L, c)$, where u'_i is the i-th timestep. From this, we can deduce that after one layer, the forward Mamba calculates

$$y_{L+1} = c' = C\overline{A}^{L+1}\overline{B}u'_0 + C\overline{A}^{L}\overline{B}u'_1 + \cdots + C\overline{B}c \tag{3}$$

and the backwards Mamba calculates

$$c' = C\overline{A}^{L+1}\overline{B}u'_L + C\overline{A}^{L}\overline{B}u'_{L-1} + \cdots + C\overline{B}c. \tag{4}$$

Note that the backwards Mamba layer maintains the classification token at the end of the sequence. From both layers, we now have two sequences: $y_{\text{forward}} \in \mathbb{R}^{L+1 \times D}$ and $y_{\text{backward}} \in \mathbb{R}^{L+1 \times D}$. To combine these into one sequence, we concatenate on the channel dimension and then project back to the original input shape. The final equation for one passthrough of a bidirectional block is

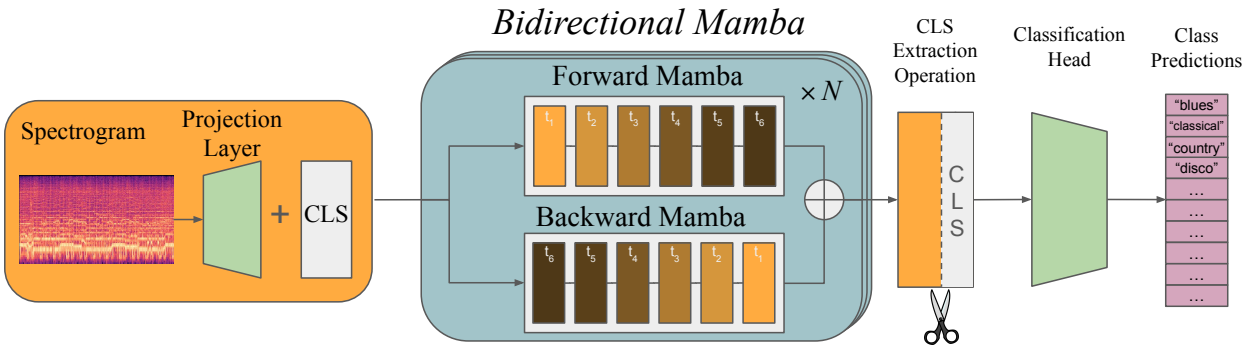

Figure 1: An overview of MoMamba. First, the spectrogram is projected to a higher dimension, followed by the concatenation of the class token. Then, these features pass through the Mamba layers. Finally, the class token is extracted and fed through the classification head. The final output of the model is the class probability distribution.

$$y_{\text{bidirectional}} = \text{Linear}(\text{Concat}((y_{\text{forward}}, y_{\text{backward}})). \tag{5}$$

After passing through all the bidirectional Mamba layers, we obtain $y_{\text{output}} \in \mathbb{R}^{L+1\times D}$. Then, we select only the classification token index c', which contains the accumulation of the information from the input sequence. The resulting classification token vector is then projected to the number of classes.

3.2.2 Mamba (SSM) Layers

After the embedding, the transformed data is passed through a stack of Bidirectional Mamba blocks, shown in the middle group in Figure 1. Each block leverages state-space modeling to capture both short- and long-term dependencies (Gu & Dao, 2024). Following each block, Layer Normalization (Ba et al., 2016) is applied to stabilize the input distribution for the subsequent block, and a Dropout layer (Hinton et al., 2012) is used for regularization. This block structure is repeated N times, where N is the number of Mamba blocks, thereby increasing the model's depth and capacity to learn complex temporal patterns.

3.2.3 Bidirectional Mamba Block

In each Bidirectional Mamba block, the model processes the sequential data in both forward and backward directions. The classification token remains at the end of the sequence, allowing it to aggregate information from both directions. The outputs from the forward and backward passes are concatenated and passed through a linear layer to project back to the input shape.

3.2.4 Classification/Regression Head

After passing through the Mamba layers, we extract the final timestep of the output sequence to get the classification token and project it to match the class size using a fully connected layer. This step ensures that the extracted representations are mapped to the appropriate number of output classes, allowing the model to make accurate predictions.

For regression, we use a two-layer feedforward head with ReLU and dropout, projecting the embedding to a single scalar via a sigmoid.

4 Experiments

4.1 Data Processing

For each input audio sample, we generate a mel-spectrogram using *Librosa*'s (McFee et al., 2025) methods to capture its frequency content over time. We generate the log-mel spectrograms with 128 mel bands using a 60 ms Hann window and a 10 ms hop size at a sampling rate of 44.1 kHz, matching the spectrogram parameters of (Huang et al., 2022), as we found this worked best with our method. The STFT was computed with an FFT size equal to the window length, and the resulting mel spectrograms were converted to the log scale (dB) with the maximum power as the reference.

4.2 Model Setup and Parameters

To prepare for training, we tune the learning rate, the embedding dimension size, and the number of bidirectional Mamba blocks. The embedding dimensions and the number of bidirectional Mamba blocks we searched through are $\{256, 512, 1024\}$ and $\{3, 4, 5, 6\}$, respectively. We find that the number of parameters is generally around 40 million for the best setup in each task.

4.3 Training and Testing

For each downstream task, we train each model from scratch. We also use data augmentation techniques to increase the number of samples in our training set. For each task, the specifics of the techniques will be described in the following paragraphs. For classification tasks, we use cross-entropy loss, and for regression tasks, we use mean squared error loss. For all models, we use AdamW (Loshchilov & Hutter, 2019) as our optimizer. To maintain data integrity, we carefully ensure that all augmented versions of a sample remain within the same split, preventing any potential data leakage and preserving the validity of our evaluations.

4.4 Baseline Methods

We select multiple baseline models for comparisons. MusicResNet (Elbir & Aydin, 2020)is a CNN-based model that was evaluated on GTZAN. However, they do not use the fault-filtered split resulting in high scores. DenseNet (Bian et al., 2019) has been shown to perform well on MIR tasks. The CRNN (Choi et al., 2017; GitYCC, 2019) combines CNNs and RNNs, resulting in promising performance. We also compare to LSTM (Hochreiter & Schmidhuber, 1997) as an additional baseline. AST (Gong et al., 2021) has been shown to utilize transformers effectively for audio tasks. AudioMamba (Erol et al., 2024) has shown that Mamba can be applied to speech and sound effects tasks. We train and evaluate each of these models from scratch on all of the following MIR problems. We also hyperparameter-tune the models to evaluate their performance under our setup.

4.5 Datasets and Downstream Tasks

To evaluate these models, we use seven different downstream tasks: meter classification, genre classification, global key classification, emotion regression for valence and arousal, instrument classification, and pitch classification. We use the following commonly used datasets:

Meter2800 (MTR2800)
We use the Meter2800 dataset (Abimbola et al., 2023) to classify the meter of a sample, which includes 1200 samples each of 3-meter and 4-meter, and 200 samples each of 5-meter and 7-meter. Every sample is 30 seconds long. The samples are taken from various raw audio datasets, one example being GTZAN (Tzanetakis & Cook, 2002). We pitch shift each sample in the training set by $[-1, 0, 1]$ semitones and time-stretch by a $[0.75, 1, 1.25]$ multiplier, resulting in 9 augmented samples for each original sample.

GTZAN
We use the GTZAN dataset (Tzanetakis & Cook, 2002), a widely used benchmark for music genre classification. However, we are aware of the well-documented issues with this dataset, including mislabeled samples, duplicates, and potential distortions (Sturm, 2014). To mitigate these concerns and ensure a more reliable

evaluation, we adopt the train-test split proposed by (Kereliuk et al., 2015), which aims to address some of these limitations. The dataset consists of 1000 30-second audio samples, evenly distributed across 10 distinct genre classes, making it a balanced dataset for classification tasks.

GiantSteps
We use the GiantSteps dataset, containing 604 samples, (Knees et al., 2015) for testing and the GiantSteps-MTG (Korzeniowski & Widmer, 2017) dataset, containing 1486 samples, for training to follow the same setup as (Korzeniowski & Widmer, 2017) for global key classification. We augment the training set by pitch-shifting each sample to the other 11 keys, improving the model's ability to generalize for this downstream task. When evaluating this task, we use the weighted score proposed by (Raffel et al., 2014), which takes into account the relationship between the predicted and true key and mode of a given sample.

EmoMusic
For emotion regression tasks, we use the EmoMusic Dataset (Soleymani et al., 2013), which contains the mean valence and arousal score for 744 45-second samples. We use the given train-test split. For this task, we concatenate the chromagram to the input spectrogram. Without concatenating the chromagram to the spectrogram, we found that many models fail to capture the harmonic content necessary for learning meaningful valence representations. Additionally, we normalize the labels to be in the range $[0, 1]$.

NSynth
We use the NSynth dataset (Engel et al., 2017), (305,979 samples), consisting of multiple pitches and instruments. It also contains synthetic and acoustic versions of each instrument. Each sample is monophonic and is 4 seconds in length. We did not use any data augmentation for this dataset. We use the given train-test splits to allow direct comparison to existing benchmarks.

4.6 Ablations on Model Design

We conduct a series of ablations to study the impact of different architectural design choices. Specifically, we evaluate the model with and without a class token, vary the placement of the class token, test the inclusion and removal of positional encodings, and compare unidirectional versus bidirectional variants. These ablations allow us to isolate how each component contributes to performance and provide insights into which design elements are most critical for capturing musical structure.

Model Setup	**NSynth$_P$ (Acc.)**
MoMamba	0.937
+ Positional Encoding	0.887
- Class Token	0.835
+ Class Token (front of sequence)	0.017
+ Class Token (middle of sequence)	0.906
Unidirectional	0.903

Table 1: Ablation study on MoMamba design choices. Performance is measured on NSynth Pitch classification (Accuracy).

Adding positional encoding to the model does not improve model performance. Bidirectionality improves the performance of Mamba by a significant margin. Finally, placing the class token at the front prevents effective aggregation of information, as reflected in the ablation study. More ablations are located in the appendix (A.1).

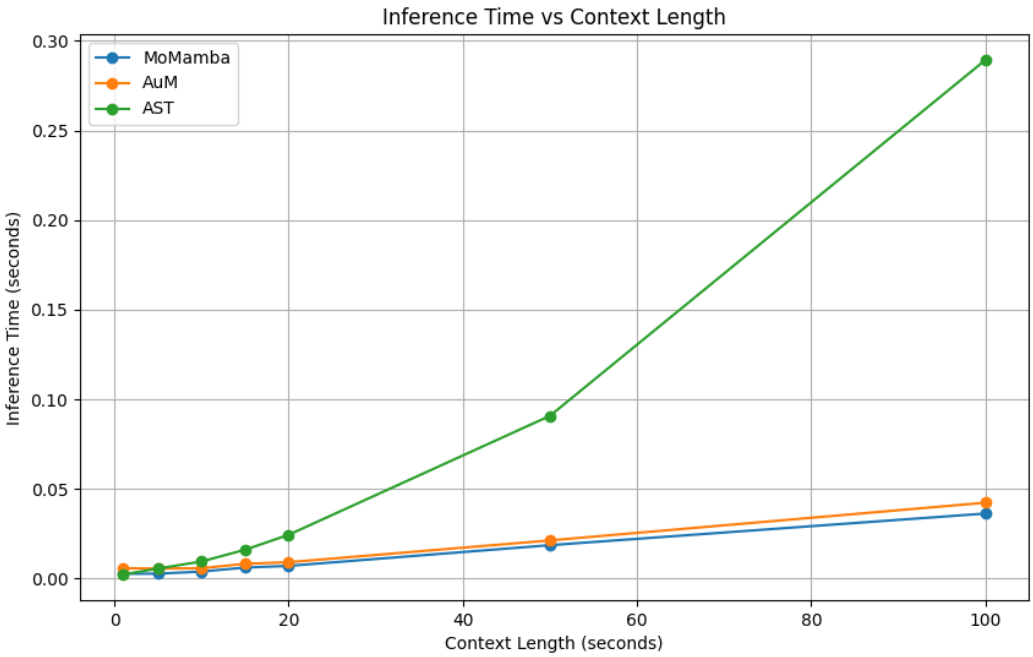

Figure 2: Inference time comparison across models.

5 Results

5.1 Analysis and Evaluation of MoMamba

The results of MoMamba and other non-pretrained models presented in Table 2, with comparisons to pretrained models reported separately in Table 3. Across all tasks, MoMamba consistently outperforms other non-pretrained models and achieves competitive performance relative to pretrained models. Considering the aggregated results, MoMamba exhibits a significantly higher average performance than all other non-pretrained models. It demonstrates particularly strong results on note-based classification tasks while maintaining comparable performance on more subjective tasks, such as emotion prediction. These findings suggest that MoMamba's architecture effectively captures both local and long-range musical dependencies without relying on large-scale pretraining, and enabling it to represent both objective structural information and higher-level perceptual attributes of music.

5.2 Inference Time Comparison of MoMamba and Other Models

We report inference time comparisons on a subset of tasks in the appendix (5). As expected, MoMamba and AudioMamba (Erol et al., 2024) achieve significantly faster inference than AST (Gong et al., 2021), while remaining slower than lightweight CNN-based models such as MusicResNet (Elbir & Aydin, 2020) and CRNN (GitYCC, 2019). Mamba-based models process full sequences with linear-time complexity, enabling them to capture long-range dependencies more efficiently than transformers, but at the cost of some raw speed compared to highly parallelized CNNs.

In summary, MoMamba provides an attractive middle ground for music classification: it delivers strong performance with substantially lower inference cost than transformers, while retaining the ability to capture global structure that CNNs often miss.

Dataset	**MTR2800**	**GTZAN**	**GS**	**EMO_V**	**EMO_A**	**NSynth_P**	**NSynth_I**	**Average**
Metric	Acc.	Acc.	W. Acc.	R2	R2	Acc.	Acc.	
MusicResNet 11	0.694	0.619	0.304	–*	–*	0.206	0.016	—
DenseNet 21; 4	0.808	0.433	0.218	0.159	0.283	0.239	0.091	0.319
CRNN 14; 6	0.741	0.646	0.138	0.060	0.370	0.906	0.734	0.514
LSTM 20	0.672	0.366	0.302	0.120	0.427	0.745	0.719	0.479
AST 15	0.694	0.617	0.253	0.228	**0.549**	0.884	0.743	0.567
AudioMamba 13	0.650	0.567	0.451	0.064	0.193	0.897	0.716	0.505
MoMamba	**0.827**	**0.667**	**0.556**	**0.355**	0.538	**0.937**	**0.774**	**0.665**

Table 2: Accuracy metrics across multiple tasks for non-pretrained models: Meter Detection (MTR2800), Genre Classification (GTZAN), Key Estimation (GS), Emotion Valence (EMO_V), Emotion Arousal (EMO_A), NSynth Pitch (NSynth_P), and NSynth Instrument (NSynth_I). Bolded numbers indicate the best performance in each column. The Average column shows the mean performance across all tasks, considering only models with results available for every task. * indicates that this model fails to perform on the task.

5.3 Limitations of Mamba

MoMamba achieves faster inference compared to RNNs and Transformers due to how SSMs operate, which allows it to process sequences efficiently without the sequential dependencies of RNNs or the quadratic complexity of self-attention in Transformers. Unlike RNNs, which require sequential updates, Mamba computes updates in parallel, significantly reducing latency. Additionally, it avoids the expensive attention mechanism of transformers, making it more scalable for long sequences. Despite these advantages, CNNs offer better inference speed. CNNs leverage localized convolutions and highly parallelized matrix operations, making them extremely efficient for structured data like spectrograms. While Mamba offers a strong balance of speed and long-range modeling capability, it trades some inference speed for improved performance on tasks requiring deeper contextual understanding, making it a better choice when capturing global dependencies is more important than raw efficiency.

5.4 Discussion

MoMamba's performance across tasks highlights its strength in capturing long-range dependencies, a crucial advantage over CNN-based models that rely on localized feature extraction. Unlike convolutional architectures, which struggle to represent global musical structures, Mamba effectively learns key transitions and harmonic relationships over extended sequences, shown through the downstream task performance.

While transformers have shown impressive results in music modeling, they require extensive pretraining on large datasets to perform well. Training a transformer from scratch often leads to suboptimal results due to its reliance on self-attention mechanisms, which demand vast amounts of data to generalize effectively (Gong et al., 2021; Copet et al., 2023; Dhariwal et al., 2020). In contrast, MoMamba demonstrates strong performance even without pretraining, suggesting that its sequence modeling approach is inherently more data-efficient. Moreover, its linear computational complexity makes it scalable for long musical sequences compared to the quadratic cost of self-attention in Transformers. These findings reinforce Mamba's potential for real-time music analysis and suggest that future work could further enhance its capabilities through large-scale pretraining.

We reiterate that MoMamba cannot be compared directly against large pretrained models; we are not aware of any SSM pretrained on large music datasets. Future large-scale training of a Mamba model will enable such comparisons.

Model	GTZAN	GS	EMO_V	EMO_A	NSynth_p	NSynth_I
MoMamba	0.667	0.556	0.355	0.538	<u>0.937</u>	<u>0.774</u>
MERT-95M$^{\text{K-means}}$ (Li et al., 2024)	0.786	0.650	0.529	0.699	0.713	0.746
MERT-95M-public$^{\text{K-means}}$ (Li et al., 2024)	0.728	<u>0.673</u>	0.597	0.725	0.704	0.756
MERT-330M$^{\text{RVQ-VAE}}$ (Li et al., 2024)	<u>0.793</u>	0.656	<u>0.612</u>	<u>0.747</u>	**0.944**	0.726
Musicset-Sup (McCallum et al., 2022)	**0.835**	0.286	0.566	0.726	0.793	0.731
Audioset-Sup (McCallum et al., 2022)	0.748	0.667	0.341	0.545	0.819	0.676
Musicset-ULarge (McCallum et al., 2022)	0.735	0.210	0.577	0.700	0.892	0.740
Audioset-ULarge (McCallum et al., 2022)	0.672	0.287	0.438	0.624	0.805	0.721
Musicset-USmall (McCallum et al., 2022)	0.686	0.508	0.389	0.668	0.824	0.714
Audioset-USmall (McCallum et al., 2022)	0.797	0.197	0.386	0.609	0.777	0.698
MuQ (1 codebook) (Zhu et al., 2025)	0.828	0.459	0.553	0.746	–	–
MuQ (4 codebooks) (Zhu et al., 2025)	0.783	0.651	0.590	0.737	–	–
MuQ (8 codebooks) (Zhu et al., 2025)	0.834	0.658	0.597	**0.762**	–	–
SOTA Excluding Above	–	**0.743** [26]	**0.617**[9]	0.721 [5]	–	**0.782** [36]

Table 3: Comparison of MoMamba and related models versus state-of-the-art models across various music-related tasks. While direct comparisons to state-of-the-art pretrained models are inherently asymmetric due to their extensive pre-training, we include them to provide context against current standards for these tasks. Bolded numbers represent the best in the task and underlined represent the second best.

5.5 Comparison to AudioMamba

MoMamba achieves superior performance compared to AudioMamba (Table 2). A key distinction lies in how the two models handle input structure: MoMamba processes the spectrogram as a continuous sequence, thereby preserving the natural temporal order of the data. In contrast, AudioMamba applies patchification, which disrupts global ordering and requires positional encodings to recover sequence information. Our ablations (Table 1), as well as the original AudioMamba design (Erol et al., 2024), show that positional encodings do not improve and may possibly be detrimental to the performance of Mamba-based models on MIR tasks. Moreover, MoMamba has half the number of parameters as AudioMamba while acheiving superior performance. These results suggest that maintaining the full spectrogram structure is more effective than patch-based representations for MIR tasks.

5.6 Comparisons to Pre-Trained Models

MoMamba demonstrates strong performance across multiple music-related tasks, notably outperforming the 95 million parameter MERT models on most benchmarks. This highlights its efficiency despite having fewer parameters than some larger pretrained alternatives. MoMamba is competitive to the pretrained models from (McCallum et al., 2022), and surprisingly outperforms every Musicset and Audioset model on the NSynth tasks. MoMamba also matches with the MERT (Li et al., 2024) models on the NSynth tasks. To make a fair comparison, we would compare a pretrained Mamba-based spectrogram model to the existing pretrained benchmarks, however, this is out of the scope of this study and left for future work. Overall, MoMamba proves to be an efficient model that delivers competitive performance despite training on small amounts of data.

6 Conclusion

State space models, specifically the Mamba model, are highly adept at capturing temporal dynamics and complex patterns in sequential musical data, making them highly effective for downstream MIR tasks. The Mamba architecture's selective scan mechanism enhances predictive accuracy and generalization, addressing key challenges in music analysis. These results underscore its potential to serve as a versatile framework for advancing MIR research. Future work may explore pretraining Mamba models on large datasets to compare with foundational models.

References

J. Abimbola, D. Kostrzewa, and P. Kasprowski. Music time signature detection using resnet18. *J Audio Speech Music Proc*, 30, 2024. doi: 10.1186/s13636-024-00346-6.

Jeremiah Abimbola, Daniel Kostrzewa, and Paweł Kasprowski. Meter2800: A novel dataset for music time signature detection. *Data in Brief*, 51, 2023. ISSN 2352-3409. doi: 10.1016/j.dib.2023.109736.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Wenhao Bian, Jie Wang, Bojin Zhuang, Jiankui Yang, Shaojun Wang, and Jing Xiao. Audio-based music classification with densenet and data augmentation. In Abhaya C. Nayak and Alok Sharma (eds.), *PRICAI 2019: Trends in Artificial Intelligence*, pp. 56–65, Cham, 2019. Springer International Publishing. ISBN 978-3-030-29894-4.

Rodrigo Castellon, Chris Donahue, and Percy Liang. Codified audio language modeling learns useful representations for music information retrieval. *arXiv preprint arXiv:2107.05677*, 2021.

Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pp. 2392–2396. IEEE, 2017.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. *Advances in Neural Information Processing Systems*, 36:47704–47720, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression, 2022. URL https://arxiv.org/abs/2210.13438.

A. Elbir and N. Aydin. Music genre classification and music recommendation by using deep learning. *Electronics Letters*, 56:627–629, 2020. doi: 10.1049/el.2019.4202.

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders, 2017.

Mehmet Hamza Erol, Arda Senocak, Jiu Feng, and Joon Son Chung. Audio mamba: Bidirectional state space model for audio representation learning. *IEEE Signal Processing Letters*, 2024.

GitYCC. crnn-pytorch: Convolutional recurrent neural network (crnn) for image-based sequence recognition using pytorch, 2019. URL https://github.com/GitYCC/crnn-pytorch. Accessed: 2025-03-11.

Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*, 2021.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Carlos Hernandez-Olivan and Jose R. Beltran. Music composition with deep learning: A review, 2021. URL https://arxiv.org/abs/2108.12290.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

Zi Huang, Shulei Ji, Zhilan Hu, Chuangjian Cai, Jing Luo, and Xinyu Yang. Adff: Attention based deep feature fusion approach for music emotion recognition, 2022. URL https://arxiv.org/abs/2204.05649.

Corey Kereliuk, Bob L. Sturm, and Jan Larsen. Deep learning and music adversaries. *IEEE Transactions on Multimedia*, 17:2059–2071, 2015. URL https://api.semanticscholar.org/CorpusID:3619742.

Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR'15)*, Málaga, Spain, October 2015.

Junghyun Koo, Gordon Wichern, François G Germain, Sameer Khurana, and Jonathan Le Roux. Understanding and controlling generative music transformers by probing individual attention heads. *arXiv preprint arXiv:2403.12345*, 2024.

Filip Korzeniowski and Gerhard Widmer. End-to-end musical key estimation using a convolutional neural network. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 966–970. IEEE, 2017.

Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghao Xiao, Chenghua Lin, Anton Ragni, Emmanouil Benetos, Norbert Gyenge, Roger Dannenberg, Ruibo Liu, Wenhu Chen, Gus Xia, Yemin Shi, Wenhao Huang, Zili Wang, Yike Guo, and Jie Fu. Mert: Acoustic music understanding model with large-scale self-supervised training, 2024. URL https://arxiv.org/abs/2306.00107.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Matthew C McCallum, Filip Korzeniowski, Sergio Oramas, Fabien Gouyon, and Andreas F Ehmann. Supervised and unsupervised learning of audio representations for music understanding. *arXiv preprint arXiv:2210.03799*, 2022.

Brian McFee, Matt McVicar, Daniel Faronbi, Iran Roman, Matan Gover, Stefan Balke, Scott Seyfarth, Ayoub Malek, Colin Raffel, Vincent Lostanlen, Benjamin van Niekirk, Dana Lee, Frank Cwitkowitz, Frank Zalkow, Oriol Nieto, Dan Ellis, Jack Mason, Kyungyun Lee, Bea Steers, Emily Halvachs, Carl Thomé, Fabian Robert-Stöter, Rachel Bittner, Ziyao Wei, Adam Weiss, Eric Battenberg, Keunwoo Choi, Ryuichi Yamamoto, CJ Carr, Alex Metsai, Stefan Sullivan, Pius Friesch, Asmitha Krishnakumar, Shunsuke Hidaka,

Steve Kowalik, Fabian Keller, Dan Mazur, Alexandre Chabot-Leclerc, Curtis Hawthorne, Chandrashekhar Ramaprasad, Myungchul Keum, Juanita Gomez, Will Monroe, Viktor Andreevitch Morozov, Kian Eliasi, nullmightybofo, Paul Biberstein, N. Dorukhan Sergin, Romain Hennequin, Rimvydas Naktinis, beantowel, Taewoon Kim, Jon Petter Åsen, Joon Lim, Alex Malins, Darío Hereñú, Stef van der Struijk, Lorenz Nickel, Jackie Wu, Zhen Wang, Tim Gates, Matt Vollrath, Andy Sarroff, Xiao-Ming, Alastair Porter, Seth Kranzler, Voodoohop, Mattia Di Gangi, Helmi Jinoz, Connor Guerrero, Abduttayyeb Mazhar, toddrme2178, Zvi Baratz, Anton Kostin, Xinlu Zhuang, Cash TingHin Lo, Pavel Campr, Eric Semeniuc, Monsij Biswal, Shayenne Moura, Paul Brossier, Hojin Lee, Waldir Pimenta, Jon Petter Åsen, Shin Hyun, Iliya S, Eugene Rabinovich, Geo Lei, Jize Guo, Phillip S.M. Skelton, Matt Pitkin, Anmol Mishra, Slava Chaunin, BenedictSt, Scott VanRavenswaay, and David Südholt. librosa/librosa: 0.11.0rc1, February 2025. URL https://doi.org/10.5281/zenodo.14908061.

Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. Mir_eval: A transparent implementation of common mir metrics. In *ISMIR*, pp. 367–372, 2014. URL http://dblp.uni-trier.de/db/conf/ismir/ismir2014.html#RaffelMHSNLE14.

Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, April 2018. ISSN 2192-662X. doi: 10.1007/s13735-018-0154-2. URL http://dx.doi.org/10.1007/s13735-018-0154-2.

M. Soleymani, Michael N. Caro, Erik M. Schmidt, Cheng-Ya Sha, and Yi-Hsuan Yang. 1000 songs for emotional analysis of music. In *CrowdMM '13*, 2013. URL https://api.semanticscholar.org/CorpusID:15438733.

Bob L. Sturm. The state of the art ten years after a state of the art: Future research in music information retrieval. *Journal of New Music Research*, 43(2):147–172, April 2014. ISSN 1744-5027. doi: 10.1080/09298215.2014.894533. URL http://dx.doi.org/10.1080/09298215.2014.894533.

G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

Luyu Wang, Pauline Luc, Yan Wu, Adria Recasens, Lucas Smaira, Andrew Brock, Andrew Jaegle, Jean-Baptiste Alayrac, Sander Dieleman, Joao Carreira, and Aaron van den Oord. Towards learning universal audio representations, 2022. URL https://arxiv.org/abs/2111.12124.

Haina Zhu, Yizhi Zhou, Hangting Chen, Jianwei Yu, Ziyang Ma, Rongzhi Gu, Yi Luo, Wei Tan, and Xie Chen. Muq: Self-supervised music representation learning with mel residual vector quantization, 2025. URL https://arxiv.org/abs/2501.01108.

A Method

A.1 Model Ablations

Model Setup	NSynth_{P} acc.	NSynth_{I} acc.	$\text{EmoMusic}_{\text{V}}$ R2	$\text{EmoMusic}_{\text{A}}$ R2
MoMamba	0.937	0.774	0.355	0.538
MoMamba (w. Unidirection)	0.903	0.660	0.192	0.479
+ Positional Encoding	0.887	0.755	0.270	0.477
- Class Token	0.835	0.738	0.036	0.481
+ Class Token (front of sequence)	0.017	0.209	0.036	0.009
+ Class Token (middle of sequence)	0.906	0.740	0.249	0.428

Table 4: Accuracy of different model setups on MIR tasks.

In addition to the ablations in section 4.6, we also tested the model setups on the NSynth instrument and EmoMusic tasks to further solidify our claims. The consistency of the results further show the robustness of the model setup. From this, we see that bidirectional Mamba with a class token at the end of the sequence performs the best out of every setup.

The design of AudioMamba involves placing the classification token in the middle of the input patchified spectrogram. The problem of not incorporating the entire spectrogram is addressed (Erol et al., 2024) when placing the classification token in the middle of the sequence. The bidirectionality ensures that the whole sequence is "seen" by the classification token, the first half by the forward direction and the second half by the backward direction. However, with MoMamba, we find that placing the classification token at the end of the sequence results in better performance. This may be because AudioMamba uses learnable positional embeddings, which, when the spectrogram is patchified, could misrepresent the temporal order and limit the information the classification token receives.

B Experiment Details

B.1 Inference Time

Model	MTR2800	GTZAN	GS
MusicResNet (Elbir & Aydin, 2020)	**0.73 ± 6.41**	**0.59 ± 9.1**	**2.69 ± 18.28**
DenseNet (Huang et al., 2016; Bian et al., 2019)	11.23 ± 8.68	5.18 ± 5.12	7.84 ± 23.42
CRNN (GitYCC, 2019; Choi et al., 2017)	1.79 ± 5.56	1.39 ± 4.16	3.85 ± 19.43
LSTM (Hochreiter & Schmidhuber, 1997)	72.91 ± 1.99	70.64 ± 7.71	16.72 ± 6.70
AST (Gong et al., 2021)	495.67 ± 25.88	64.98 ± 3.60	6.44 ± 17.95
AudioMamba (Erol et al., 2024)	10.34 ± 7.18	15.58 ± 8.70	5.01 ± 24.12
MoMamba	6.11 ± 6.92	9.91 ± 2.45	3.29 ± 12.11

Table 5: Average inference times (ms) with standard deviations for three tasks: Meter Detection (MTR2800), Genre Classification (GTZAN), and Key Estimation (GS). Bold numbers indicate the fastest inference in each column. Each model's inference time was recorded using a batch size of 32 and same data processing to avoid any confounding factors. Each model was evaluated on a 48 GB L40 GPU.

B.2 Data Visualization for EmoMusic

Below are the visualizations for both valence and arousal regression tasks for each non-pretrained model on the test set. The R2 scores for each visualization are the values in table 2 Note that AST does not learn to predict valence well even if it's score is not the lowest, as it learns to predict the average score every time. The red line indicates perfect predictions.

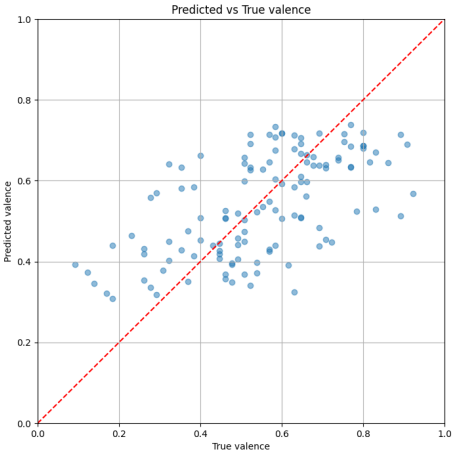

Figure 3: MoMamba Valence

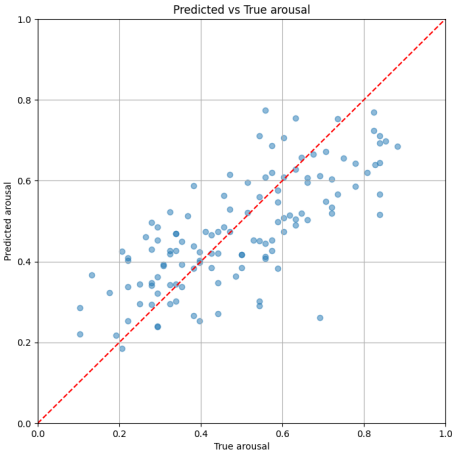

Figure 4: MoMamba Arousal

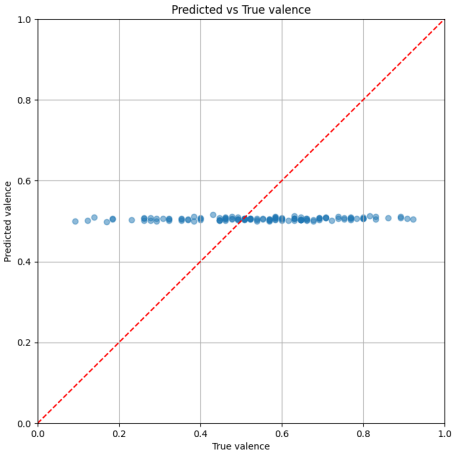

Figure 5: AST Valence

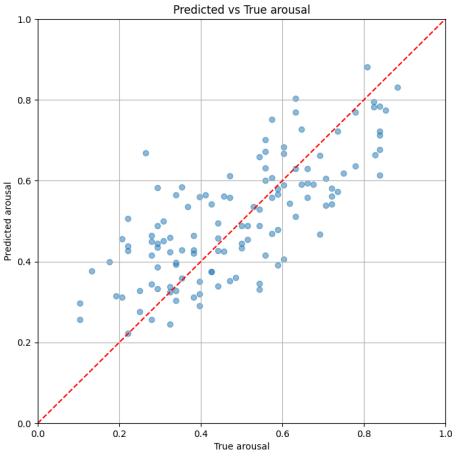

Figure 6: AST Arousal

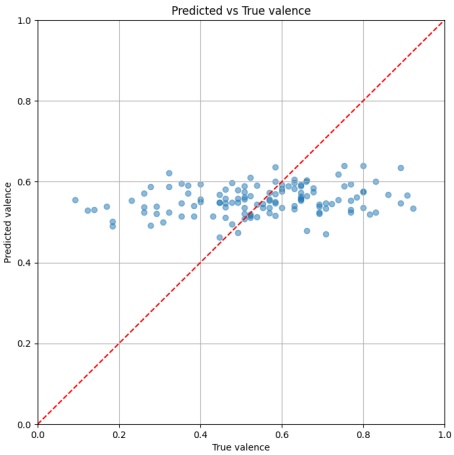

Figure 7: CRNN Valence

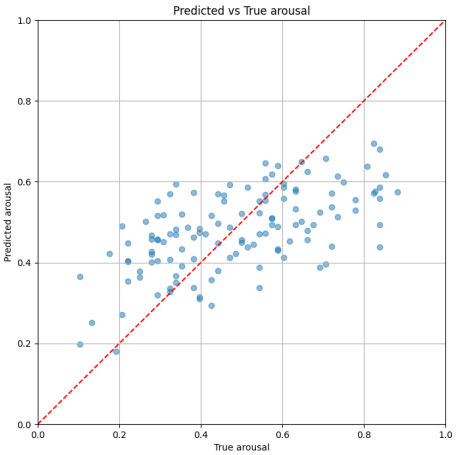

Figure 8: CRNN Arousal

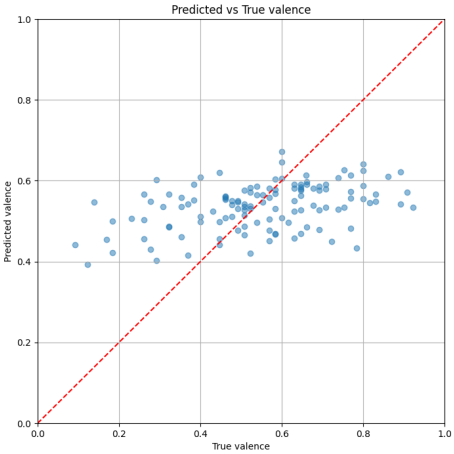

Figure 9: Densenet Valence

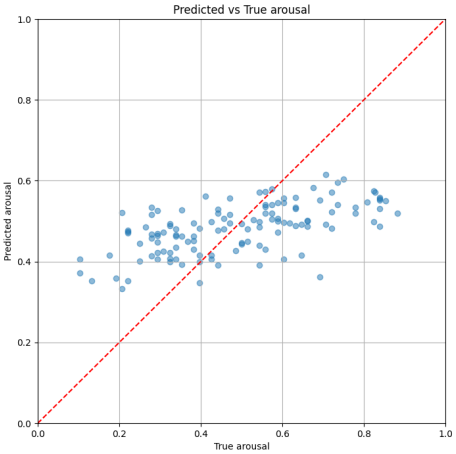

Figure 10: Densenet Arousal

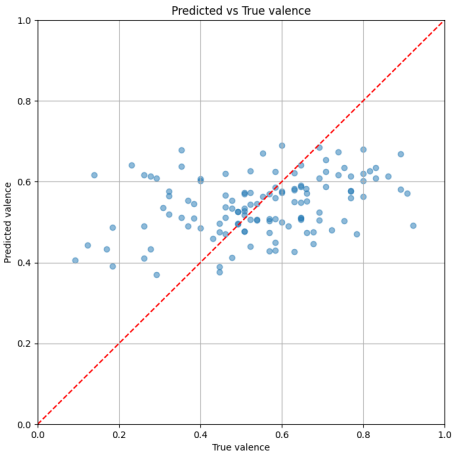

Figure 11: LSTM Valence

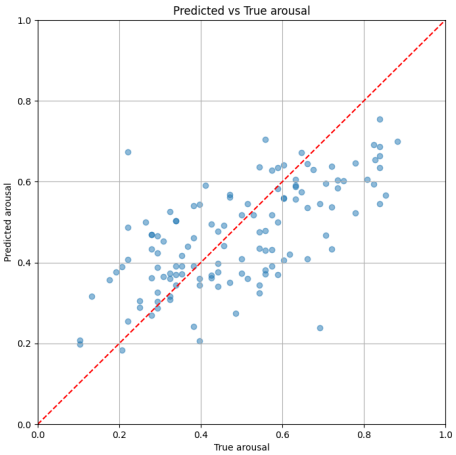

Figure 12: LSTM Arousal

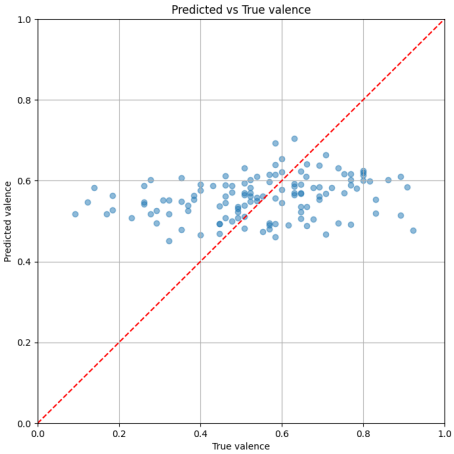

Figure 13: AudioMamba Valence

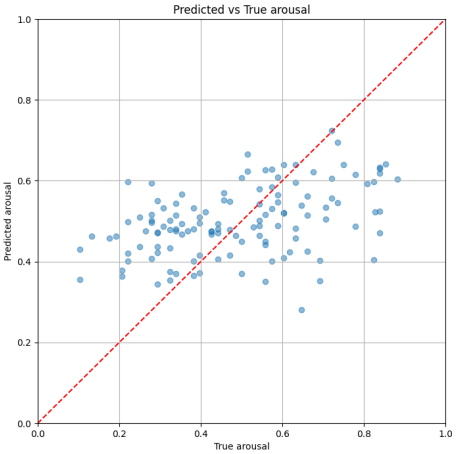

Figure 14: AudioMamba Arousal