Efficient Text-to-Music Generation via Flow Matching with Bidirectional Mamba SSM

1st Anthony Wang
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA, USA
anw061@ucsd.edu

2nd Shlomo Dubnov
Dept. of Music & Dept. of Computer Science and Engineering
University of California, San Diego
La Jolla, CA, USA
sdubnov@ucsd.edu

***Abstract*—In this paper, we describe in detail our approach used to create our submission to the ICME 2026 Grand Challenge on Academic Text-to-Music Generation. This competition challenges participants to develop a generative text-to-music model in a controlled environment with constraints to ensure that results are easily reproducible. Participants can submit to the Efficiency and/or Performance Track. For our design, we take advantage of the recent advances in the area of state-space models (SSM), which have allowed them to attain similar performance to transformers without the bottleneck of quadratic memory growth in the standard attention mechanism. This makes them a useful alternative for tasks that would typically rely on a transformer, especially for researchers that lack the time or resources to train large transformer-based architectures. We present a lightweight model based on flow-matching with an SSM that achieves strong semantic alignment with the text prompt while remaining within the parameter restrictions of the Efficiency Track.**

***Index Terms*—Text-to-music generation, generative AI, flow-matching, state-space model (SSM)**

I. Introduction

The ICME 2026 Grand Challenge on Academic Text-to-Music Generation is a competition aimed to increase transparency and encourage innovation in architecture design for text-to-music systems [3]. The current state-of-the-art systems are often proprietary designs with billions of parameters such as MusicGen backed up by difficult to obtain datasets and extensive computing resources not easily accessible, thus limiting the ability of researchers in academia to innovate in this area. Thus, this challenge aims to address this issue by having all participants train exclusively on the publicly available MTG-Jamendo music dataset as well as having a track specifically limiting the number of parameters that can be used in the core generative model, thus eliminating advantages provided by special datasets and encouraging the exploration of efficient models.

The model we developed for this competition relies on state-space models (SSM) at its core. A key advantage of SSMs is their memory and time efficiency, which grows linearly with respect to the input sequence length instead of the quadratic computational complexity of the attention mechanism [7]. Additionally, recent innovations in SSMs have allowed them to become a highly capable, efficient alternative to transformers in Large-Language Model (LLM) applications, providing equal or better performance at a much lower computational cost [2]. Furthermore, preliminary work on applying SSMs in text-to-music generation have demonstrated its potential to be a suitable alternative to transformers [4]. For our architecture, we utilize an SSM with the flow-matching algorithm [5]. The generated audio from our model using SSMs in the place of transformers indeed reinforces SSM as an efficient, viable architecture for music generation, which is discussed in further detail in the "Results" section.

II. Track Participation

Since we were limited in GPU resources along with the fact we learned of and entered this challenge relatively late, we chose to focus on the Efficiency Track. This track limits the number of trainable parameters in the core generative model to ≤ 500M parameters, which was well-suited to our computational and time bottlenecks. Our final model turned out to have 189M parameters, staying well below the limit. With the computational resources available to us, we could train this model for our intended number of steps in approximately 3 days, allowing time for debugging and retraining if necessary.

III. Methodology

A. Pre-trained resources

Previous research in generative modeling have demonstrated that working directly in "raw" data spaces is inefficient and models perform better when operating in latent spaces. For example, the architecture of Stable Diffusion uses a variational autoencoder to encode images, performing the diffusion process in the latent space instead of working with raw pixels [6]. We follow a similar approach and encode the audio files from the MTG-Jamendo dataset into latent representations instead of directly using the raw waveforms. Here, instead of an variational autoencoder, we use the music deep compression autoencoder (DCAE) ACE-Step-v1-3.5B parameter model, whose implementation is freely available in a Github repository. Internally, the encoding process first transforms an audio waveform into a mel-spectrogram with two channels before encoding it to a $(8, 16, T)$-shaped tensor, representing 8 latent channels at 10.77 Hz, which is a highly compact representation [1]. Before providing it directly as input into our flow matching network, we flatten the first two dimensions together into a $(16 \cdot 8, T) = (128, T)$-dimensional tensor so it

can be accepted as input into the SSM, as it is a sequential model.

Since our model accepts a text prompt as input, we use the FLAN-T5 base text encoder to generate text embeddings to feed into the network. In both the training and sampling (generation) process, we use FLAN-T5 to encode the text prompt prior to feeding it into the flow matching network, which will be discussed in more detail in the next section.

For both of these models, we freeze their parameters to ensure they are not changed during training. Thus, they do not contribute to the total parameter count for the core generative model as per the grand challenge rules, allowing us to stay below the 500M parameter limit for the Efficiency Track.

B. Architecture

Our generation and training algorithm is flow matching, which has proven to be the state-of-the-art on various generation tasks especially in audio and video compared to traditional diffusion algorithms. The neural network we used consists of 24 blocks, with a bidirectional Mamba SSM block as the key component in each block. The Mamba SSM builds upon the traditional SSM setup by making the matrices mapping between the inputs, hidden representations, and outputs dependent on the input instead of keeping them constant [2]. This enables Mamba to selectively choose which inputs are relevant to updating the hidden state, providing an architecture with the same effect as the attention mechanism while bypassing the quadratic time and memory bottleneck. On language tasks, it has been shown that Mamba exceeds the performance of transformer-based models while using less parameters [2]. This makes Mamba an attractive choice given the parameter constraints in the Efficiency Track.

By default, the Mamba SSM only propagates information forward in one direction. However, musical structure is dependent on both what comes before and after the current section of music being generated. As such, to ensure global musical coherence, we use a bidirectional Mamba SSM. The bidirectional Mamba runs two independent Mamba SSM blocks with the exact same configuration, one with the input in the order it is given and the other with the input reversed representing the music being played "backwards" in time. This is analogous to the setup used in a bidirectional recurrent neural network. We then concatenate both representations together before feeding it into a single fully-connected layer so the model can learn how to map the two embeddings into a single representation.

Within each block, we incorporate standard cross-attention between the input audio latents and the T5 text embeddings. We set the latents as the queries and the text embeddings as the keys/values, providing conditioning to the model based on the prompt. Additionally, we also make use of a small multilayer perceptron with two layers to introduce non linear positional mixing after passing through the bidirectional Mamba blocks and cross-attention operations. All three components are wrapped in a residual connection as per standard deep learning practice.

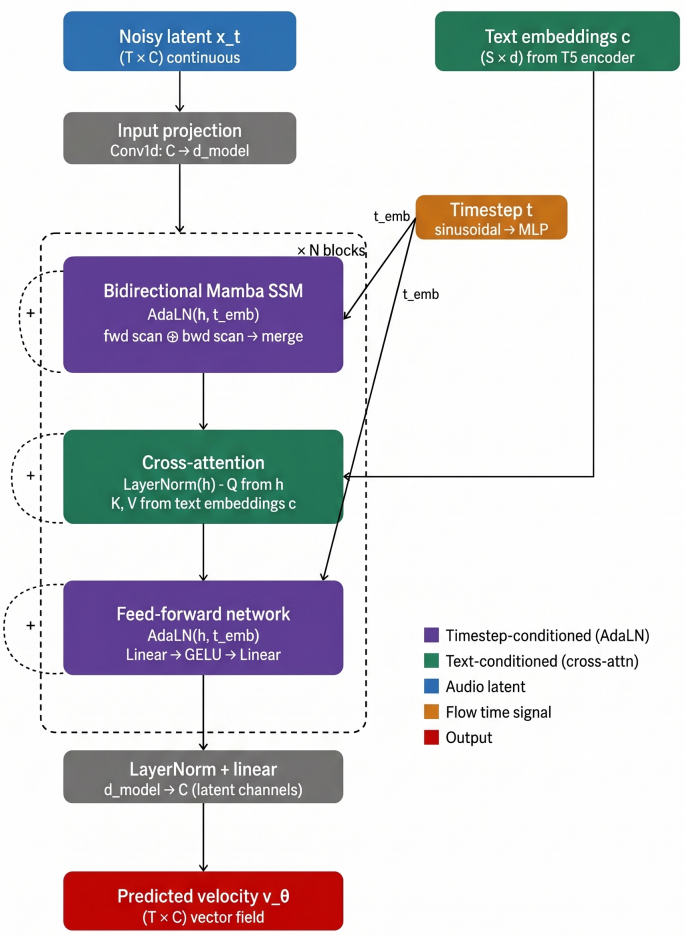

Fig. 1. Complete Flow Matching Architecture Overview.

Lastly, prior to feeding an audio latent to the 24 block neural network, we perform a projection from the flattened 128-dimensional input to the 512-dimensional hidden space of the bidirectional Mamba SSM blocks via a 1D convolution. At the end, we use a fully connected layer that maps the output from the blocks back to the original dimensions of the latents. We also compute a 256-dimensional sinusoidal timestep embedding using a 2 layer multilayer perceptron, which is standard in generative modeling. This timestep embedding is then injected into the network using a Diffusion-Transformer (DiT) style Adaptive Layer Normalization, modulating the scale and shifts of activations inside each bidirectional Mamba block and the MLP. A complete overview of the architecture can be seen in Fig. 1.

C. Training

We use the MTG-Jamendo dataset for training as per the grand challenge rules [3]. This dataset contains approximately $55,000$ files of high quality music, which were preprocessed to remove any vocals to ensure the training data is purely instrumental. We used the full dataset representing 3777 hours of music combined. We first convert these audio files into a latent representation using the ACE-Step autoencoder described

previously. Prior to doing this we estimate the sample mean and variance of the latents from a subset of the files, using it to normalize the latents so that they have approximately zero mean and unit variance.

We then proceeded to train using the flow matching with classifier-free guidance objective for $300,000$ steps [8]. We start with Gaussian noise sampled from $z \sim \mathcal{N}(0, I)$. Given a time $t \in [0, 1]$, we use a linear interpolation between z and the clean latent x:

$$x_t = (1 - t)z + tx$$

Let c be the text embeddings of the prompt from the T5 encoder and v_θ represent the neural network with parameters θ, whose output predicts the "velocity" x_t should move at to reach the the target latent x. We pass x_t, t, c to v_θ to compute the mean-squared error loss between the predicted and target velocity of $(x-z)$, the straight line distance between the noisy and clean latent:

$$\mathcal{L}_{FM} = \mathbb{E}\left[||v_\theta(x_t, t, c) - (x - z)||_2^2\right]$$

We then use this loss to compute gradients and perform backpropagation to train our flow matching network. During training, we sample the time t from a standard logit normal distribution. This distribution has the highest density around $t = 0.5$, which is beneficial since the halfway point along the probability path in flow matching represents a point where x_t is an approximately uniform mixture of noise and the true latent, so sampling frequently around $t = 0.5$ allows the model to accurately learn the velocity at the region where it is most difficult to do so.

Our training process utilized a GPU cluster provided by our institution. The most powerful GPUs available to us were the Nvidia RTX A5000 and Nvidia A30, each with 24 GB of VRAM. For most of our training, we used an A5000. Towards the end, an issue arose with the computing node that runs the A5000, forcing us to switch to an A30 to finish all $300,000$ steps. Additionally, each job running on the cluster was limited to 12 hours of runtime, which we accounted for by saving checkpoints every $5,000$ steps so we could quickly resume training once each session was terminated. In total, we spent 81 hours of GPU time to train the model. In each training session, we used an AMD EPYC 7543P CPU, utilizing 8 cores as well as 64 GB of memory.

D. Inference

We use the standard Euler's method as an ODE solver to generate a music sample once training is complete. We start with Gaussian noise sampled from $z_0 \sim \mathcal{N}(0, I)$. Let $\emptyset$ represent the null text embedding. Using a classifier-free guidance coefficient of $\sigma = 3.5$, the velocity at time t is computed as

$$s_t = v_\theta(z_t, t, \emptyset) + \sigma \cdot (v_\theta(z_t, t, c) - v_\theta(z_t, t, \emptyset))$$

to yield the update

$$z_{t_{i+1}} = z_{t_i} + \Delta t \cdot s_{t_i}$$

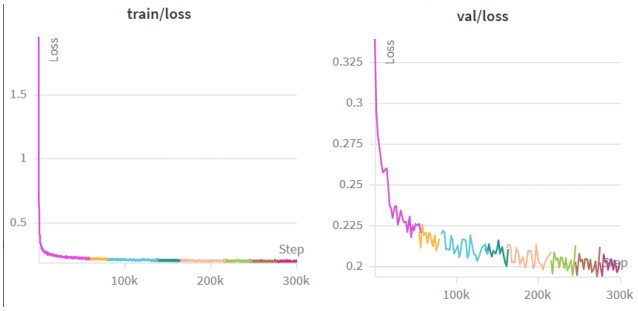

Fig. 2. Training and validation loss plots.

TABLE I
OBJECTIVE EVALUATION RESULTS [3]

Model Name	FAD ↓	CLAP ↑	CCS ↑	# of parameters
FluxAudio-S (Baseline)	0.757	0.088	0.592	120M
Stable Audio Open	0.574	0.321	0.800	1.1B
MusicGen-small	0.574	0.370	0.875	300M
MusicGen-medium	0.548	0.353	0.892	1.5B
MusicGen-large	0.553	0.379	0.888	3.3B
Ours	**0.577**	**0.338**	**0.863**	**189M**

For each music sample generated, we used $n = 100$ uniform time steps between $[0, 1)$ so that $t_i = \frac{i}{100}$ for $i = 0, 1, \ldots, 99$

IV. SUBMISSION

We made one submission using the final model weights saved after $300,000$ training steps. We ran the generation process on each of the 100 test prompts a few times with different random seeds and submitted the one that seemed to give the best FAD and CLAP scores based on local testing.

V. RESULTS

The training and validation loss curves over the course of $300,000$ steps of training are shown in Fig. 2. As seen in the plots, the training process was stable and matches the expected behavior while training a flow-matching model, with the loss rapidly decreasing over the first few thousand steps before eventually flattening out and stabilizing. The validation loss, which is computed every $2,000$ steps, remains consistent with the training loss, indicating that no overfitting is occurring.

Our submission achieved excellent results and performed comparably to other open-source and industry text-to-music models tested by the Grand Challenge organizers, which are summarized in Table I. We beat the baseline FluxAudio-S model across all three objective metrics in FAD, CLAP, and concept coverage scores (CCS) by a significant margin all while using only ∼70M additional parameters [3].

Most notably, we were able to achieve the highest CLAP score of **0.338** among all submissions to the Grand Challenge, indicating a strong semantic alignment between the generated music and the prompt. Additionally, we also achieved the second highest concept coverage score (CCS) of **0.863**, indicating that the generated audio files accurately reflects the musical concepts indicated by the prompt such as genre, instrument,

TABLE II
MEAN OPINION SCORE (MOS) RESULTS [3]

Model Name	MOS (all)	MOS (expert)
MusicGen-small	3.538	3.425
Ours	**3.225**	**3.177**

and mood/theme. Taken together, these two scores indicate that our model is able to generate music that closely adheres to the requests of the prompt. We attained second place out of 16 submissions both within the Efficiency Track and overall based on these objective metrics, beating teams on the performance track whose models had substantially more parameters than ours.

The results from the Mean Opinion Study (MOS) conducted by the Grand Challenge organizers support the objective metrics. In this phase, 35 participants (of which 20 were "expert" listeners) rated music samples from finalist submissions as well as a track from MusicGen-small on a set of prompts [3]. For each music sample, participants assigned an "Overall" rating on a 5 point Likert scale which is a holistic evaluation of its alignment with the prompt and musical quality. As can be seen in Table II, our model's subjective ratings are competitive with those attained by MusicGen-small. This suggests our objective metrics indeed correlate positively with real human perception, affirming the quality of our model. We ultimately won 2nd place out of 4 finalists in the Efficiency track based on the MOS study results.

We would also like to highlight the performance of our model compared to Stable Audio Open and MusicGen. We were able to beat Stable Audio Open in CLAP and CCS, only falling slightly short in FAD. This is despite Stable Audio Open having over $5\times$ the number of parameters as our model and being trained on 7.3K hours of training data compared to the MTG-Jamendo's 3.7K hours [3]. While we were surpassed in all three metrics by the MusicGen models, our results are still competitive, especially considering that MusicGen had the benefit of 20K hours of training data which included propriety data inside Meta. Together, these results suggest that text-to-music systems do not necessarily require a substantial number of parameters and/or training data to produce strong results on these metrics; with innovations like the Mamba SSM, similar outcomes can be realized with much smaller models quickly trainable with fairly modest computational resources.

While our model achieved highly competitive semantic alignment, we noted a higher FAD score of **0.577** compared to other submissions. This result indicates that our generated music features had a higher statistical distributional divergence from a reference dataset of real music. It highlights a known tradeoff in generative audio in which greater adherence to the prompt can cause the model's output distribution to deviate from the reference dataset baseline, a phenomenon also observed in the MusicGen results reported by the Grand Challenge organizers [3]. Finding methods to lower the FAD score while maintaining faithfulness to the prompt remains an excellent area for further investigation.

VI. CONCLUSION

We have presented a generative text-to-music model that achieves strong semantic alignment using flow-matching with a bidirectional Mamba SSM consisting of 189M parameters. The objective results obtained with this model was able to perform just as well and beat other submissions containing significantly more parameters, demonstrating that careful architecture selection can allow a relatively lightweight model to obtain competitive results. We hope that these results encourage other researchers to further explore efficient methods in music generation that do not require access to substantial computing resources and propriety training data.

ACKNOWLEDGEMENTS

We would like to thank the University of California, San Diego's Data Science and Machine Learning Platform (DSMLP) for providing us access to the GPUs, CPUs, memory, and storage necessary to successfully compete in the Grand Challenge.

REFERENCES

[1] Junmin Gong, Sean Zhao, Sen Wang, Shengyuan Xu, and Joe Guo. Ace-step: A step towards music generation foundation model, 2025. URL: https://arxiv.org/abs/2506.00045, arXiv:2506.00045.

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

[3] Fang-Chih Hsieh, Wei-Jaw Lee, Chun-Ping Wang, Hung-yi Lee, Hao-Wen Dong, and Yi-Hsuan Yang. Academic text-to-music grand challenge: Datasets, baselines, and evaluation methods. In *International Conference on Multimedia and Expo, Grand Challenge Paper*, 2026.

[4] Wei-Jaw Lee, Fang-Chih Hsieh, Xuanjun Chen, Fang-Duo Tsai, and Yi-Hsuan Yang. Exploring state-space-model based language model in music generation, 2025. URL: https://arxiv.org/abs/2507.06674, arXiv:2507.06674.

[5] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL: https://arxiv.org/abs/2210.02747, arXiv:2210.02747.

[6] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL: https://arxiv.org/abs/2112.10752, arXiv:2112.10752.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL: https://arxiv.org/abs/1706.03762, arXiv:1706.03762.

[8] Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky T. Q. Chen. Guided flows for generative modeling and decision making, 2023. URL: https://arxiv.org/abs/2311.13443, arXiv:2311.13443.