LLM-based Finetuning for Text to Symbolic Music Generation

Shih-Lun Wu, Josephine Lee, and Sofronie Dun

{slseanwu, jleey, sofronie}@mit.edu

6.7960 (F24) Deep Learning Final Project

1. Introduction

Symbolic music generation models (Huang et al., 2018; Wu and Yang, 2023; Thickstun et al., 2024) generate music as sequences of notes played by one or many instruments. Compared to the audio-domain counterparts (Agostinelli et al., 2023; Copet et al., 2023), which produce continuous waveforms of multi-instrument mixtures, symbolic model outputs are more interpretable and editable, and hence can be more seamlessly integrated into musicians' composition workflows.

However, an important conditioning modality, namely text prompting, is missing in current symbolic music generation models. This is in contrast to text-to-audio music models (Agostinelli et al., 2023; Copet et al., 2023) due potentially to the relative scarcity of paired (text, symbolic music) data samples. The absence of text control means that users have to condition the model on individual notes and/or chords, which could be a slow and tedious process. Existing works that finetune (or repurpose) text LLMs for symbolic music tackled simple outputs like melody lines and chord progressions (Yuan et al., 2024; Deng et al., 2024), which are musically much less rich and diverse than multi-instrument compositions.

Fortunately, a paired (text, symbolic music) dataset, MidiCaps (Melechovsky et al., 2024), featuring 170K songs, was compiled recently—the authors leveraged LLMs to expand musical tags (e.g., keys, moods, genres, and chords) extracted from LMD dataset (Raffel, 2016) symbolic music samples to natural text descriptions. We believe that this new dataset provides a great avenue to explore methods to finetune either (unconditional) symbolic music models (Thickstun et al., 2024), or text LLMs (Grattafiori et al., 2024), which have no specialized musical skills but are trained on orders of magnitude larger data and hence boast impressive generalizability, to perform text-to-symbolic-music generation. In addition to comparing the two fine-tuning paradigms described above, we also propose several straightforward techniques hoping to enhance the inductive biases for both fine-tuning (or adaptation) paradigms.

Contributions

We propose and evaluate two main approaches for the task of text-conditioned multi-track symbolic music generation:

1. **Using Llama 3.2 LLM representations to enable text conditioning of pretrained unconditional symbolic music model**
    a. We integrate the Llama 3.2 text LLM (Grattafiori et al., 2024) with the Anticipatory Music Transformer (AMT) model (Thickstun et al., 2024) to enable text-conditioned multi-track symbolic music generation, using the (frozen) Llama text representations at the last layer as conditions to AMT.
    b. We propose a novel variant where the fine-tuned AMT model can learn weighted importances over Llama's text representations at different layers, each of which may capture different levels of abstraction (e.g., semantics or high-level motifs). We expect that these layer-wise representation weights facilitate more nuanced mapping from text conditions to symbolic music outputs, and hence enhance text controllability.

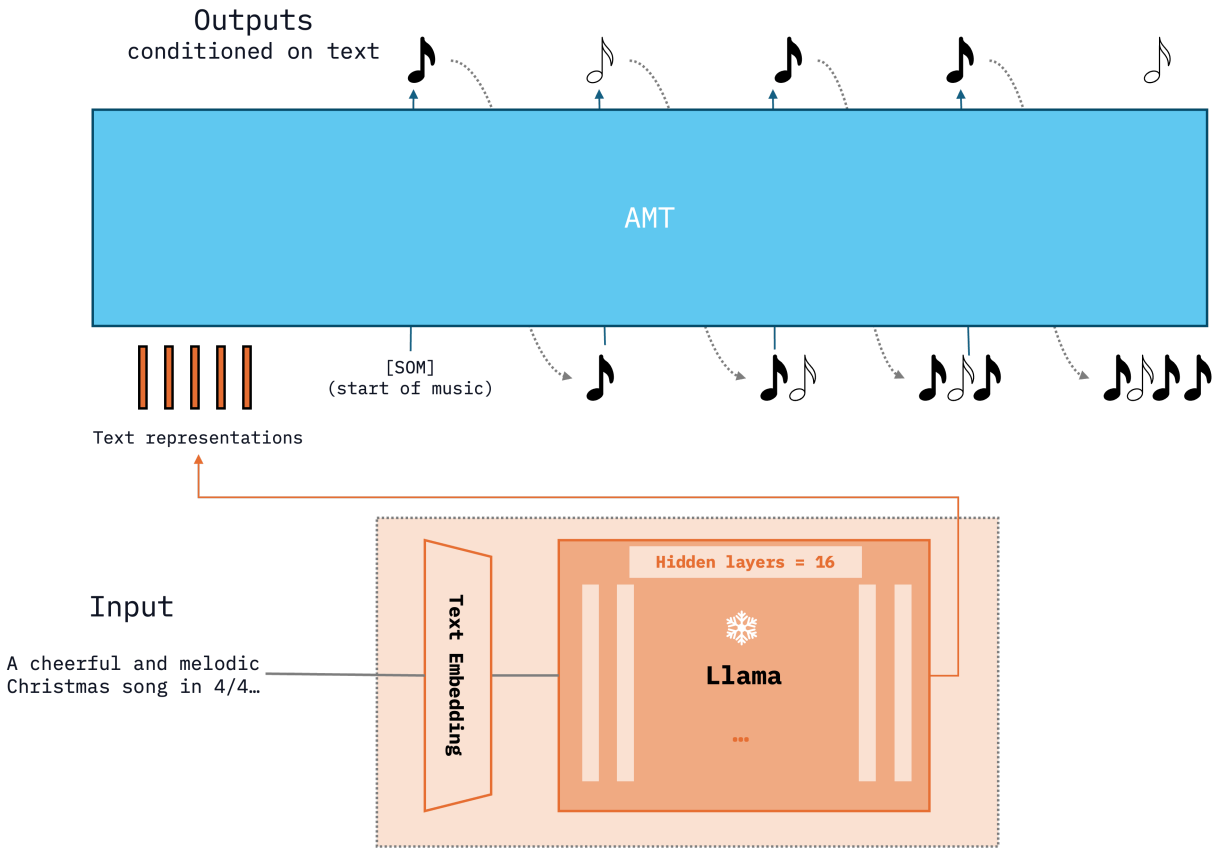


Figure 1: Enabling text control on a pretrained unconditional music generation model, namely, the Anticipatory Music Transformer (AMT), leveraging a frozen Llama 3.2 text LLM.

2. **Finetuning Llama 3.2 text LLMs to repurpose them as a text to symbolic music generation model**
    a. We leverage AMT's music tokenization scheme and expand the vocabulary (i.e., input embeddings and output projection) of Llama for it to take and generate music tokens.
    b. As an attempt to imbue Llama with better inductive biases on music, we additionally initialize expanded music vocabulary embeddings using pretrained embeddings from the AMT model. This approach leverages the *learned musical meanings* of AMT embeddings while capitalizing on Llama's generalization capabilities. We hope that pretrained music embeddings help the model better capture the relationship between music tokens at the start of fine-tuning, and hence make it easier for the model to learn the nuances between text and music tokens.

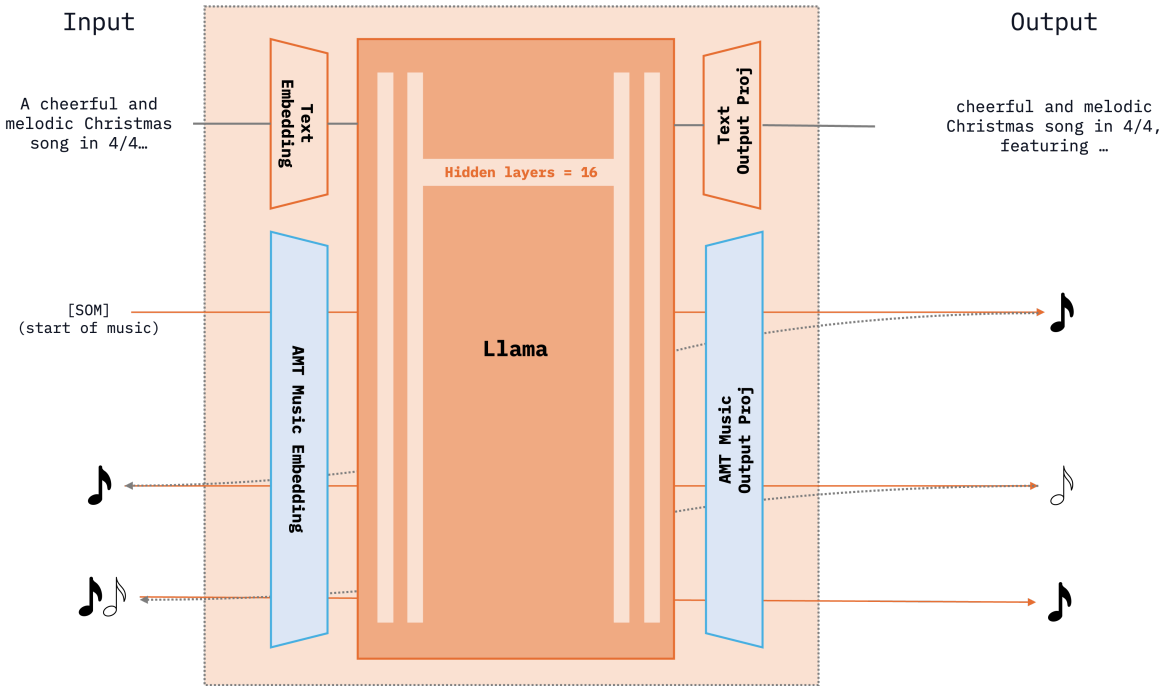


Figure 2: Repurposing the Llama LLM for conditional symbolic music generation, by expanding its vocabulary to handle symbolic music tokens.

2. Related Work

2.1. Text-to-music generation

Text-to-music generation involves creating music based on textual descriptions. For example, *MusicLM* (Agostinelli et al., 2023) leverages hierarchical sequence-to-sequence modeling to produce high-fidelity music conditioned on text descriptions. The model further supports melody-conditioned music

generation, where a provided melody is transformed to match a specific stylistic description. Whereas *MusicGen* (Copet et al., 2023) simplifies the generation process by introducing interleaving techniques to model multiple streams of compressed music representations based on the *EnCodec* (Défossez et al., 2022) audio tokenizer. This single-stage transformer-based approach enhances the efficiency of text-to-music generation and achieves competitive results in both human and automatic evaluations.

One critical drawback is their reliance on waveform outputs, which are inherently less editable. This restricts the ability to make fine-grained modifications to the generated music, a crucial feature for many music creators. Additionally, while MusicGen improves efficiency, it lacks fine-grained control over how closely the generated output adheres to the conditioning text or melody. These limitations emphasize the need to explore symbolic music generation approaches that provide more flexibility and control over musical elements such as harmony, rhythm, and texture.

2.2. Adapting LLMs to generate symbolic music

Symbolic music generation offers an alternative to waveform-based approaches by enabling the creation of editable representations, such as MIDI files. Models like *ChatMusician* (Yuan et al., 2024) and *ComposerX* (Deng et al., 2024) adapted Large Language Models (LLMs) for symbolic music tasks by translating musical notes to text strings in, for example, ABC notation (wiki). These models perform well in generating coherent melodies and structured compositions (e.g., melodies paired with lyrics and/or chord progressions) using pretrained LLMs fine-tuned (or few-shot prompted) with textual music representations. However, their focus on relatively simple melodic structures falls short of capturing the richness and complexity typical of human-composed music. This gap underscores the importance of developing techniques that can model intricate harmonic progressions, rhythmic patterns, and dynamic textural details found in multi-instrument compositions.

3. Technical Background

3.1. Tokenization scheme for multi-track symbolic music

We use the specialized tokenization scheme for multi-track symbolic music proposed in the Anticipatory Music Transformer (AMT) paper (Thickstun et al., 2024), encoding musical notes as context-free triplets: `(arrival time, instrument+pitch, note duration)` . That is, one note played by any instrument would become 3 tokens in the sequence.

We present a summary of the vocabulary below.

Table 1: AMT's token vocabulary for multitrack symbolic music

Category	Tokens	Vocab Size
Arrival Time	Represents when a note begins; quantized time intervals (10 ms steps, maximum 100 seconds)	10,000
Instrument+Pitch	Combines the MIDI instrument identifier with the pitch of the note. There are 129 instruments and 128 possible pitches for each instrument.	16,512

Category	Tokens	Vocab Size
Note duration	Quantized durations for each note (in 10ms steps, from 10ms to 10sec)	1,000
Total	Unique tokens for all categories combined	27,512

3.2. Paired text descriptions for music

The MidiCaps dataset (Melechovsky et al., 2024) combines symbolic music (MIDI files) with descriptive text captions, enabling the exploration of relationships between musical content and natural language. The text description for each music composition is generated following the process below.

1. **Music Feature Extraction:** The pipeline extracts several music-specific features from MIDI files using music information retrieval (MIR) tools (Choi et al., 2017). These features include tempo (the pace of the piece), chord progression, time signature, instrument presence, genre, and mood.
2. **Transformation into Natural Language**: After extracting these features, the pipeline uses a large language model (LLM), specifically Anthropic's Claude 3, to generate descriptive captions. By leveraging in-context learning, the LLM transforms the extracted symbolic features into fluent natural language sentences, capturing the music's essence in human-readable form. This approach effectively bridges MIR and natural language processing, creating a versatile dataset for tasks like text-to-MIDI generation and music captioning.

4. Method

4.1. Starting from (unconditional) music model (AMT)

We add the following network components to the existing music model, namely, the AMT model (Thickstun et al., 2024):

- **Llama 3.2 (1B) model integration:** A pre-trained Llama 3.2 (1B) model is loaded and frozen to provide rich contextual embeddings for text conditioning. The weights are frozen and there is a specialized padding token to handle batched training and inference. A learnable linear layer projects the Llama hidden states into the AMT embedding space.
- **Concatenation with music tokens**: The projected Llama text embeddings are prepended, along the timestep dimension, to the AMT token embeddings before entering the AMT backbone. Additionally, We ensure that the Llama text embeddings, which already encode positional information inside Llama, do not receive AMT's positional embeddings to avoid redundant (and potentially misleading) positional information.

We propose two versions of AMT models:

- **Base Version**: The base model directly uses the final hidden state from Llama as the conditioning input to AMT, following MusicGen (Copet et al., 2023).

- **Enhanced Version:** All Llama hidden states (from 16 Transformer layers and the embedding layer) are stacked into a tensor and combined as a weighted sum. The weights are learnable parameters, enabling the model to learn and decide the importance of individual layers dynamically during training.

The loss function for next-token prediction is defined as the negative log-likelihood of the target token given the input sequence. We denote x as the music tokens, y as the text tokens, and T as the music token sequence length.

$$-\log p(x|y) = -\sum_{t=1}^{T} \log p(x_t|y, x_{<t}) \,.$$

At the beginning of all music tokens, we place a `[start of music]` token to signal the model that text description has ended and that it should start generating music.

4.2. Starting from text model (Llama 3.2)

We experiment with both the 1B and 3B version of Llama 3.2 and add the following network components to the existing Llama text model to enable it to generate music.

- **Expanded Vocabulary Handling:** We add an embedding layer for the extended music vocabulary and a linear projection layer to align the embeddings of the extended vocabulary with the pretrained representation space of Llama. The extended vocabulary is exactly the same as that of AMT.
- **Prefix Instruction:** Inspired by instruction fine-tuning (Wei et al., 2022), which aligns model's behavior with user input specification, we prepend to the text description a short instruction `You are a world-class composer. Please compose some music according to the following description:` to *nudge* the LLM into a musician's mindset.

We propose two types of initialization:

- **Base Version:** the extended vocabulary embeddings are initialized following the Llama model's standard initialization method, i.e., gaussian random initialization.
- **Enhanced Version:** For the extended vocabulary embeddings, weights are initialized by copying from a pretrained AMT's token embeddings, and scaling them such that the standard deviation of the embeddings match that of the Llama's pretrained text token embeddings, and finally, zeros are filled for the dimensions non-existent in AMT (AMT has token embedding dimension of 1280, while Llama has either 2048 or 3072, depending on the model size).

For tokens in the original vocabulary, the model predicts based on `lm_head`, which maps hidden states to logits corresponding to the original Llama (text) vocabulary. For tokens in the extended vocabulary, `expanded_lm_head` produces logits corresponding to the extended music vocabulary. These logits are finally concatenated with those from the original vocabulary to produce a single output distribution over all text and music tokens.

Unlike in AMT, we apply cross-entropy loss over both the text and music vocabularies as its loss function, written as:

$$-\log p(x, y) = -(\sum_{t=1}^{T} \log p(x_t | y, x_{<t}) + \sum_{t'=1}^{T_{\text{text}}} \log p(y_{t'} | y_{<t'}))\,,$$

where T_{text} is the number of tokens in the text prompt. The `[start of music]` token is placed the same way as in our AMT models.

5. Experiments

5.1. Implementation and hyperparameters

Table 2: Model / training / inference hyperparameters

Attribute \ Model	**AMT Large**	**Llama 3.2 (1B)**	**Llama 3.2 (3B)**
Trainable params	783 M	1.47 B	3.56 B
Optimizer	AdamW	AdamW	AdamW
Training seq length	1024	1024	1024
Training batch size	16	64	64
Peak learning rate	1e-5	2e-4	2e-4
Max training steps	60 K	15 K	15 K
Warmup (linear) steps	500	500	500
Learning rate schedule	cosine	cosine	cosine
Precision	bf16-mixed	bf16-mixed	bf16-mixed
Inference temperature	1.0	1.0 / 1.1	1.0 / 1.1
Inference top-p	0.98	0.98	0.98

Training

We implement our changes based on HuggingFace `GPT2LMHeadModel` (for AMT) and `LlamaForCausalLM` (for Llama) model classes, and leverage HuggingFace `Trainer` to train models.

Following both AMT and Llama papers, we use the *AdamW* (Loshchilov and Hutter, 2019) optimizer for training. Following AMT's pretraining setup, we restrict our training sequence length to 1024 tokens, which equates to about 15 seconds of music (but highly variable due to varying note density). For the key training hyperparameters, we perform a small-scale search across:

- *batch size* = {16, 64}
- *learning rate* = {1e-5, 2e-4},

and find that AMT trains better with *small batch size* (i.e., 16) and *low learning rate* (i.e., 1e-5), while Llama, in contrast, favors the exact opposite. Maximum training (i.e., gradient update) steps is set to 60K and 15K respectively for AMT and Llama so that the models would see around 1B tokens in total, roughly equal to the token count of our training dataset (but is not 1 epoch, to be explained in Section 5.2). A summary of the hyperparameters can be found in Table 2.

We train each of the models on a single NVidia A100 GPU with 80GB of graphics memory, and we accumulate the gradients until reaching the specified batch sizes above. It takes around 1 day to train the Llama 3.2 (1B) model, and roughly 2 days for the Llama 3.2 (3B) and AMT Large models. The AMT Large model trains slower, despite its smaller size, due to not having a valid *FlashAttention 2* (Dao, 2023) implementation.

Inference

To generate samples for evaluation, we condition the models with a text prompt and ask them to generate 1024 music tokens, staying consistent with how the models are trained. For Llama models, we utilize HuggingFace's `model.generate()` method. For the AMT model, due to `model.generate()` not being compatible with additional text conditioning, we implement our own generation loop.

Following AMT's inference setup, we use *nucleus sampling* (Holtzman et al., 2020) with a *top-p* hyperparameter of 0.98. However, for Llama models, this setting sometimes leads to degenerate samples, e.g., unreasonably many notes at the same arrival time, and the arrival time never advances — we find that applying a higher temperature, i.e., 1.1 instead of the default 1.0, largely resolves the issue with the tradeoff that the outputs sound more chaotic.

With batched generation, generating 1K samples takes around **30 / 40 / 60** minutes for the **Llama (1B) / Llama (3B) / AMT Large** models respectively, also on a single A100 (80G) GPU.

As our evaluation metrics take audio inputs (and it's also easier for humans to judge the outputs with audios rather than a sequence of tokens), we utilize the *FluidSynth* to synthesize the music tokens into audio waveforms.

5.2. Dataset and data processing

We perform an intersection between the valid MIDI files in the *LMD* dataset (**Raffel, 2016**), and those which are paired with a text description in the *MidiCaps* dataset (Melechovsky et al., 2024), available on HuggingFace `datasets`, to construct our paired (text, music) dataset. This step leaves us with 151,526 samples (i.e., songs) in total.

Following the splitting strategy described in the AMT paper (Thickstun et al., 2024), we split the samples to:

- `train` set — LMD hex ID starting with `[0-9|a-d]`, 132,460 samples
- `valid` set — LMD hex ID starting with `e`, 9,618 samples
- `test` set — LMD hex ID starting with `f`, 9,448 samples

Considering limited time and compute, we use only the first **4K** and **1K** samples in `valid` and `test` sets respectively. Since the samples have vastly different duration, and hence the token sequence length, we divide each sample into non-overlapping chunks of 1,024 tokens (i.e., the same as our training sequence length), which gives us around 1B tokens in total.

Since there is only one text description per sample (i.e., multiple chunks in the same sample correspond to the same text description), when looping through the dataset, we treat one epoch as looping through all the text descriptions. That is, if one sample has four chunks, the `[1st, 2nd, 3rd, 4th, 1st, …]` chunks get fed to the model in the `[1st, 2nd, 3rd, 4th, 5th, ...]` epoch. Given our maximum training steps (15K with batch size 64, or, 60K with batch size 16), we are training our models for around 7 epochs.

5.3. Evaluation metrics

We follow conventions from recent text-to-music research (Agostinelli et al., 2023, Copet et al., 2023) and adopt the two metrics below, measuring **quality** and **text controllability** respectively.

Fréchet audio distance (FAD)

FAD evaluates the **quality** (or, precisely, the **realisticness** compared to ground-truth human compositions) of the set of all generated samples as a whole (Kilgour et al., 2019). It leverages a pretrained audio feature extractor (e.g., an audio event classifier) to obtain embeddings from each generated (and ground truth) sample, estimate the gaussian distribution parameters (i.e., means and covariances) from the set of all generated/ground-truth audio embeddings, and compute the Fréchet distance between two gaussian distributions, i.e.,

$$\mathbf{F}\left(\mathcal{N}_b, \mathcal{N}_e\right) = \left\|\mu_b - \mu_e\right\|^2 + \operatorname{tr}\left(\Sigma_b + \Sigma_e - 2\sqrt{\Sigma_b \Sigma_e}\right) ,$$

where $\mathcal{N}_e\left(\mu_e, \Sigma_e\right)$ are estimated gaussian distribution from the *generated* set embeddings, and $\mathcal{N}_b\left(\mu_b, \Sigma_b\right)$ are that from the *ground-truth* set embeddings.

CLAP score

CLAP score measures the **relevance** of the generated music **w.r.t. the input text description**. It utilizes contrastively pretrained dual audio and text encoders, i.e., CLAP (Wu et al., 2023), to extract the text embedding (of an input text description) and the audio embedding (of the generated music). Then, it computes the cosine similarity between text and audio embeddings:

$$\text{CLAPScore}(X_a, X_t) = \cos(E_a, E_t)$$

where X_a is the audio (synthesized from the generated music), X_t is the input text description, $E_a = \text{CLAP}_{\text{audio}}(X_a)$ is the audio embedding, which is the CLAP audio encoder features, normalized to be on a unit hypersphere, and $E_t = \text{CLAP}_{\text{text}}(X_t)$ is the text embedding, i.e., CLAP text encoder features, also normalized to be on a unit hypersphere.

6. Results and Discussions

6.1. Quantitative results

Training and validation losses

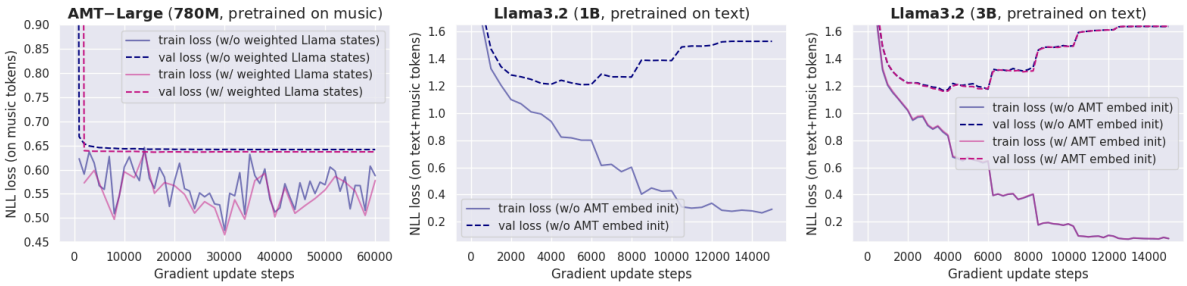


Figure 3: Training and validation losses of our models. Note that the losses of AMT and Llama models are not directly comparable since the latter models both text and music tokens.

We first examine how the training process goes for each model by observing the losses, which are plotted in Figure 3.

For the **AMT-Large** models, both *with* or *without* weighted text representations from the Llama 3.2 (1B) LLM, validation losses converge very quickly within the first 5K steps. Training losses continue to decrease slowly throughout the 60K steps, but there is no sign of serious overfitting. The quick convergence of validation loss potentially suggests the pretrained AMT backbone is not really trying to make use of the text conditioning signals, and still mostly relies on the music context, which it has extensively seen during pretraining, to predict the next tokens. The variant with weighted Llama states does score a slightly lower (better) validation loss. We take a look at the trained weights (to combine text representations of all Llama layers) and observe that the model assigns an almost uniform weights to all layers, indicating that the model finds a simple combination of all layers' representations can be helpful in optimizing the loss, instead of only using the last layer as done in prior works (Copet et al., 2023, Evans et al., 2024).

The training dynamics for the **Llama 3.2 LLMs**, in contrast, is clearly distinct from that of AMT. Both 1B and 3B versions are able to easily overfit our training dataset (with an order 100K text and symbolic music pairs), and the validation loss starts to worsen at only around 6K steps, which is around 3 epochs over the training set. We take this as a good sign as it indicates that text LLMs generally have no difficulty fitting music data, even though the entire music token vocabulary is new. Besides, our AMT embedding initialization technique, which we experiment with the 3B version, does not have a significant effect on the training and validation losses.

Evaluation of generated music

*Table 3: Evaluation results on 1K generated samples, conditioned on the first 1K descriptions of our test set. To quantify FAD's uncertainty, we compute **FAD-sub** by randomly drawing 5 sets of 500 samples from the generated 1K samples and compute FAD for each set. Standard error of mean (SEM) follows ±.*

Model \ Metrics	**Temperature**	**Val loss (↓)**	**FAD (↓)**	**FAD-sub (↓)**	**CLAP (↑)**
Baselines					
Ground truth	–	–	–	–	0.238 ± 0.003
Ground truth (random text)	–	–	–	–	0.158 ± 0.003
AMT-Large (unconditional)	1.0	–	0.781	0.833 ± 0.025	0.120 ± 0.003
AMT-Large + text cond					
w/o weighted Llama states	1.0	0.642	0.299	0.332 ± 0.035	0.156 ± 0.003
w/ weighted Llama states	1.0	0.637	0.302	0.334 ± 0.021	0.153 ± 0.003
Llama LLMs					
1B w/o AMT embed init	1.0	1.211	0.322	0.365 ± 0.016	0.200 ± 0.003
1B w/o AMT embed init	1.1	1.211	0.555	0.547 ± 0.077	0.209 ± 0.003
3B w/o AMT embed init	1.0	1.176	0.383	0.423 ± 0.030	0.202 ± 0.003
3B w/o AMT embed init	1.1	1.176	0.271	0.331 ± 0.008	0.210 ± 0.003
3B w/ AMT embed init	1.0	1.181	0.342	0.401 ± 0.017	0.203 ± 0.003
3B w/ AMT embed init	1.1	1.181	**0.245**	**0.280** ± 0.018	**0.215** ± 0.003

All the quantitative evaluation results are included in the table above. To better position the performance of our models, we come up with a few **baselines:**

- **Ground truth:** We measure the CLAP score between (text, music) pairs in the test set. This is meant to represent the **best achievable text control** of our trained models.
- **Ground truth (random text)**: Similar to above, but we randomize the text descriptions in the test set. This gives us a sense of what CLAP score is achievable with **no text control**, when the music is **human-composed**.

- **AMT-Large (unconditional)**: We generate music with the unconditional AMT-Large model, and then pair the generated samples with texts in our test set. This tells us what **machine-generated** music with **no text control** would do on CLAP score — intuitively, if our models are achieve higher CLAP scores than this, it suggests that text conditioning is working to some extent.

For apples-to-apples comparisons between all models, we now set aside the Llama LLM rows with generations using temperature 1.1 (i.e., those with a brown background). We first focus on comparing the two major model families, i.e., AMT vs. Llama. If we examine the FAD scores, we can see that **AMT-Large + text conditioning** performs better, scoring an FAD-sub of around 0.33 vs. 0.36~0.42 by Llama LLMs, which is a decently meaningful gap considering the standard errors. On the other hand, on CLAP score, **Llama LLMs** outperform the AMT models by a clear margin (i.e., 0.20 vs. 0.15), and all the models are in between the unconditional baseline (0.12) and paired ground truth (0.24), demonstrating different levels of text conditioning performance.

Therefore, our general **takeaways** on the comparison between starting with a pretrained **music** or **text** backbone is that:

- Starting with a **music** model (i.e., AMT-Large) would give us **great generated music**, which is reasonable as music is its pretraining domain, but the **text control is relatively weak**.
- Starting with a **text LLM** (i.e., Llama 3.2) leads to **more effective text control**, but the **generated music quality is not as good**.

Given that controllability is a crucial aspect for generative AI tools to be useful, we would recommend starting with text LLMs and further explore techniques to enhance the music quality.

Now we dive into more fine-grained comparisons over our proposed **techniques on providing better inductive biases** for either AMT or Llama. We can see that on AMT-Large models, the use of weighted Llama text representations does not improve either the FAD or CLAP score. On the other hand, initializing the music token embeddings from pretrained AMT, which we experiment with the Llama 3B model, does improve FAD scores slightly, suggesting that this technique helps with the generated music quality.

Lastly, we examine the effects with an **increased temperature** from 1.0 to 1.1 during generation, which we try on Llama 1B and 3B models. While both metrics improve in general with the higher temperature (except for FAD on the 1B model), upon listening to the samples, we find the music to be a lot more chaotic with little repetition structures typically found in human compositions. We suspect that the improved scores are due to better instrumentation coverage, i.e., the models are more willing to generate notes played by more instruments specified in the text prompt, and that the ML model-based FAD and CLAP metrics are more sensitive to instrumentation than to whether the musical content is sensible and well-structured. However, the exact mechanisms are to be investigated.

6.2. Qualitative study

We listen to generated samples from the following two models:

- **AMT-Large w/ weighted Llama states**

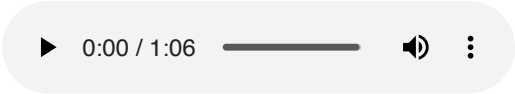
- **Llama 3.2 (3B) w/ AMT embedding initialization**

and include some of the generated music and the paired input text descriptions. For clarity, we mark parts in the text description that are being **followed** in the music in **green**, and those being **violated/ignored** in **red**. The findings largely match those from our quantitative evaluation.

The AMT model, pretrained on music, exhibits great capabilities in generating high-quality music, through rich arrangements and well-tensioned, well-developed musical ideas. However, as can be seen from the large portions of text marked in red, ineffective text control is its significant drawback.

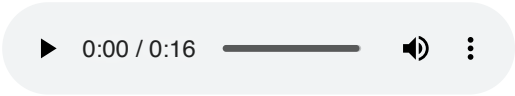
In contrast, the 3B Llama model, pretrained on (a much larger volume of) text, boasts great text control (text descriptions mostly marked in green), getting the vibes and instrumentation correct most of the time. Yet, the music it generates are mostly simple, with significantly fewer twists and turns compared to the AMT model, and thus much less catchy. Also, when we attempt to increase the temperature (1.0 → 1.1), the outputs immediately sound more chaotic and unstructured, rather than being richer and more diverse as we desire.

AMT-Large with weighted Llama states



In this short electronic ambient piece, a synth lead takes the spotlight, accompanied by a string ensemble, synth pad, and piano. With a moderate tempo and a 4/4 time signature, it evokes a sense of spaciousness and darkness, creating a dreamy and relaxing soundscape. Set in the key of C# minor, the composition features a chord sequence of Ab, F#, E, Eb, and Ebm.

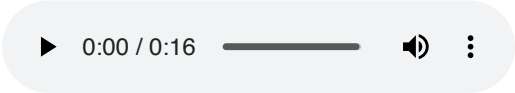
AMT Sample 01



A melodic pop instrumental song with a Christmas vibe, featuring acoustic guitar and piano leads accompanied by fretless bass, vibraphone, and drums. Set in the key of Eb major with a very fast tempo, it follows a chord progression of Ab, Abm, Eb, Ebm6, and Fm7 in a 4/4 time signature, creating a relaxing and happy atmosphere.

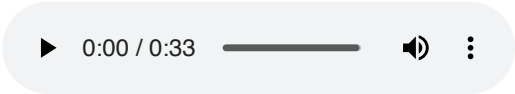
AMT Sample 02

Llama 3.2 (3B) with AMT embedding initialization, temperature = 1.0



```
A cheerful and motivational pop song with a Christmas vibe, featuring
clean electric guitar, acoustic guitar, alto saxophone, fretless bass,
and tango accordion, all blending together in a melodic harmony. Set
in the key of C# minor with a 4/4 time signature, the song maintains a
moderate tempo of 100 beats per minute, creating a pleasant and
uplifting atmosphere.
```

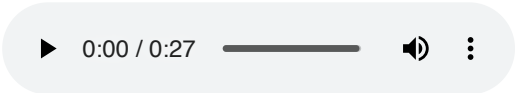
Llama Sample 01




```
A heartwarming pop song that celebrates love and happiness, featuring
a charming blend of piano, acoustic guitar, and string ensemble. With
a moderato tempo and a 4/4 time signature, this melodic ballad unfolds
in the key of G major, evoking a sense of Christmas cheer.
```

Llama Sample 02

Llama 3.2 (3B) with AMT embedding initialization, temperature = 1.1



```
A short classical piece, likely part of a film soundtrack, featuring a
string ensemble of violin, cello, and viola. With a Vivace tempo and a
4/4 time signature, it evokes a sense of drama, adventure, and
romance. The composition is in the key of C major and incorporates a
chord sequence of Fm7, G7/B, C, Ab7, and G7.
```

Llama (temp = 1.1) Sample 01

7. Limitations and Future Work

Overfitting issues

As we observed in the experiments (see Figure 3), all Llama 3.2 (1B and 3B) text LLMs exhibit serious levels of overfitting when fine-tuned on music data. This likely has resulted from a combination of too little fine-tuning data, and too many trainable parameters. As future work, on the data front, following the strategy of *Textbooks are all you need* (Gunasekar et al., 2023), we can consider adding music-domain continued pretraining data to equip the text LLM with more generalizable musical knowledge before fine-tuning on music generation. Meanwhile, on the modeling front, we can experiment with low-rank adaptation methods, e.g., LoRA (Hu et al., 2021) and QLoRA (Dettmers et al., 2023) to reduce the number of trainable parameters, and potentially scale to larger (frozen) backbones (e.g., 8B or 70B Llama models) as a stronger starting point.

Short sequence length

Due to resource constraints, we chose 1024 as our training sequence length, which is roughly only equal to 15 seconds of music. The issues with this approach are multifold:

- Some longer-term repetition structures, e.g., motivic development or verse-chorus forms, often take more than 15 seconds to unfold. With enough time and resources, we should increase the training sequence length to 4K (i.e., 1 minute) or 16K (i.e., roughly entire songs), for the model to better capture the aspects above.
- The text descriptions in *MidiCaps* (Melechovsky et al., 2024) are associated with full songs, hence are likely less accurate on crops of songs. This can be resolved if we train on full songs. However, ideally, we would prefer a model that can generate short excerpts, e.g., 10 seconds, based on text descriptions describing highly localized attributes, e.g., "`melody pitches going up with fragmented dense rhythm patterns`". To obtain such training data pairs, we can leverage audio multimodal LLMs, e.g., *Qwen2-audio* (Chu et al., 2024), to produce text descriptions for music of different durations, and apply some filtering mechanisms to ensure data quality.

References

References

Aa Name	Title	Authors
Grattafiori et al., 2024	The Llama 3 Herd of Models	Grattafiori, Aaron Dubey, Abhimanyu Jauhri, Abhinav Pandey, Abhinav Kadian, Abhishek et al.
Melechovsky et al., 2024	MidiCaps: A large-scale MIDI dataset with text captions	Melechovsky, Jan Roy, Abhinaba Herremans, Dorien
Thickstun et al., 2024	Anticipatory Music Transformer	Thickstun, John Hall, David Donahue, Chris Liang, Percy
Agostinelli et al., 2023	MusicLM: Generating Music From Text	Agostinelli, Andrea Denk, Timo I. Borsos, Zalán Engel, Jesse Verzetti, Mauro Caillon, Antoine Huang, Qingqing Jansen, Aren Roberts, Adam Tagliasacchi, Marco Sharifi, Matt Zeghidour, Neil Frank, Christian
Copet et al., 2023	Simple and Controllable Music Generation	Copet, Jade Kreuk, Felix Gat, Itai Remez, Tal Kant, David Synnaeve, Gabriel Adi, Yossi Défossez, Alexandre
Evans et al., 2024	Stable Audio Open	Evans, Zach Parker, Julian D. Carr, C. J. Zukowski, Zack Taylor, Josiah Pons, Jordi

Aa Name	Title	Authors
Deng et al., 2024	ComposerX: Multi-Agent Symbolic Music Composition with LLMs	Deng, Qixin Yang, Qikai Yuan, Ruibin Huang, Yipeng Wang, Yi Liu, Xubo Tian, Zeyue Pan, Jiahao Zhang, Ge Lin, Hanfeng Li, Yizhi Ma, Yinghao Fu, Jie Lin, Chenghua Benetos, Emmanouil Wang, Wenwu Xia, Guangyu Xue, Wei Guo, Yike
Yuan et al., 2024	ChatMusician: Understanding and Generating Music Intrinsically with LLM	Yuan, Ruibin Lin, Hanfeng Wang, Yi Tian, Zeyue Wu, Shangda Shen, Tianhao Zhang, Ge Wu, Yuhang Liu, Cong Zhou, Ziya Ma, Ziyang Xue, Liumeng Wang, Ziyu Liu, Qin Zheng, Tianyu Li, Yizhi Ma, Yinghao Liang, Yiming Chi, Xiaowei Liu, Ruibo Wang, Zili Li, Pengfei Wu, Jingcheng Lin, Chenghua Liu, Qifeng Jiang, Tao Huang, Wenhao

Aa Name	Title	Authors
		Chen, Wenhu Benetos, Emmanouil Fu, Jie Xia, Gus Dannenberg, Roger Xue, Wei Kang, Shiyin Guo, Yike
Kilgour et al., 2019	Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms	Kilgour, Kevin Zuluaga, Mauricio Roblek, Dominik Sharifi, Matthew
Wu et al., 2023	Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation	Wu, Yusong Chen, Ke Zhang, Tianyu Hui, Yuchen Nezhurina, Marianna Berg-Kirkpatrick, Taylor Dubnov, Shlomo
Wu and Yang, 2023	Compose & Embellish: Well-Structured Piano Performance Generation via A Two-Stage Approach	Wu, Shih-Lun Yang, Yi-Hsuan
Loshchilov and Hutter, 2019	Decoupled Weight Decay Regularization	Loshchilov, Ilya Hutter, Frank
Dao, 2023	FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning	Dao, Tri
Holtzman et al., 2020	The Curious Case of Neural Text Degeneration	Holtzman, Ari Buys, Jan Du, Li Forbes, Maxwell Choi, Yejin
Huang et al., 2018	Music Transformer	Huang, Cheng-Zhi Anna Vaswani, Ashish Uszkoreit, Jakob Shazeer, Noam Simon, Ian Hawthorne, Curtis Dai, Andrew M. Hoffman, Matthew D. Dinculescu, Monica Eck, Douglas

Aa Name	Title	Authors
Dettmers et al., 2023	QLoRA: Efficient Finetuning of Quantized LLMs	Dettmers, Tim Pagnoni, Artidoro Holtzman, Ari Zettlemoyer, Luke
Hu et al., 2021	LoRA: Low-Rank Adaptation of Large Language Models	Hu, Edward J. Shen, Yelong Wallis, Phillip Allen-Zhu, Zeyuan Li, Yuanzhi Wang, Shean Wang, Lu Chen, Weizhu
Gunasekar et al., 2023	Textbooks Are All You Need	Gunasekar, Suriya Zhang, Yi Aneja, Jyoti Mendes, Caio César Teodoro Giorno, Allie Del Gopi, Sivakanth Javaheripi, Mojan Kauffmann, Piero Rosa, Gustavo de Saarikivi, Olli Salim, Adil Shah, Shital Behl, Harkirat Singh Wang, Xin Bubeck, Sébastien Eldan, Ronen Kalai, Adam Tauman Lee, Yin Tat Li, Yuanzhi
Chu et al., 2024	Qwen2-Audio Technical Report	Chu, Yunfei Xu, Jin Yang, Qian Wei, Haojie Wei, Xipin Guo, Zhifang Leng, Yichong Lv, Yuanjun He, Jinzheng Lin, Junyang Zhou, Chang Zhou, Jingren

Aa Name	Title	Authors
Défossez et al., 2022	High Fidelity Neural Audio Compression	Défossez, Alexandre Copet, Jade Synnaeve, Gabriel Adi, Yossi
Choi et al., 2017	A Tutorial on Deep Learning for Music Information Retrieval	Choi, Keunwoo Fazekas, György Cho, Kyunghyun Sandler, Mark
Wei et al., 2022	Finetuned Language Models Are Zero-Shot Learners	Wei, Jason Bosma, Maarten Zhao, Vincent Y. Guu, Kelvin Yu, Adams Wei Lester, Brian Du, Nan Dai, Andrew M. Le, Quoc V.