Audio Conditioning for Music Generation via Discrete Bottleneck Features

Simon Rouard, Yossi Adi, Jade Copet, Axel Roebel, Alexandre Défossez

AUDIO CONDITIONING FOR MUSIC GENERATION VIA DISCRETE BOTTLENECK FEATURES

Simon Rouard[1,2] **Yossi Adi**[1,3] **Jade Copet**[1] **Axel Roebel**[2] **Alexandre Défossez**[4]
[1] FAIR Meta [2] IRCAM - Sorbonne Université [3] Hebrew University of Jerusalem [4] Kyutai
srouard@meta.com, alex@kyutai.org

ABSTRACT

While most music generation models use textual or parametric conditioning (e.g. tempo, harmony, musical genre), we propose to condition a language model based music generation system with audio input. Our exploration involves two distinct strategies. The first strategy, termed textual inversion, leverages a pre-trained text-to-music model to map audio input to corresponding "pseudowords" in the textual embedding space. For the second model we train a music language model from scratch jointly with a text conditioner and a quantized audio feature extractor. At inference time, we can mix textual and audio conditioning and balance them thanks to a novel double classifier free guidance method. We conduct automatic and human studies that validates our approach. We will release the code and we provide music samples on musicgenstyle.github.io in order to show the quality of our model.

1. INTRODUCTION

In the field of music generation, prior research has predominantly focused on producing brief musical segments [1,2], MIDI generation [3], while generating long and coherent waveforms (around 30 seconds) has only recently been tackled [4–6]. Specifically, most of these recent models have been designed to perform text-to-music generation, providing a fascinating tool for creators. Other types of high-level conditioning have been used in previous work such as tempo, harmony [7]. For lower-level and aligned conditioning, the authors of [5] use melody, while [8] uses chords, piano rolls, or the drum stem. However, music is hard to describe textually and the scarcity of text-music pair datasets makes it challenging to generate music in the style of a specific artist or song, since the artist is probably not represented in the training dataset. Then a common use case would be to generate music in the style of a reference segment. This gives more control to the user since they do not have to find a textual prompt that describes the music they want to generate.

In the computer vision domain, the authors of [9] introduced textual inversion to extract visual concepts that can then be used to generate new images with a text-to-image model. Given a few images (3-5) of a concept or object, one sets them as outputs of a frozen text-to-image model with a randomly initialized learnable text embedding. Backpropagating the generative model loss on the text allows to learn new "pseudowords" in the textual embedding space of the model that match the common concept depicted on the images. One can then compose this learnt pseudoword S^* in a textual prompt to generate an image of the learnt concept (for instance "a painting of S^* in the style of Picasso").

We first adapted this method by using the text-to-music model MusicGen [5], using crops of a song to depict a concept, and optimizing the cross-entropy loss of the music language model. This approach does not need to retrain a model from scratch. However, its inference is very slow since it requires hundreds of optimization steps of the textual prompt, including gradient computation through the language model, before generating music.

To tackle this issue, we present another method where we design a style conditioner module that we jointly train with a text-to-music MusicGen model [5]. This style conditioner takes a few seconds of audio and extracts features out of it. As a result this new model can generate music using two modalities as input: waveforms and textual descriptions. Our conditioning is high level even if it can retain some lower level content such as melodic patterns or rhythm. Designing this style conditioner is challenging as we need to extract enough features to have a meaningful conditioning but not too much, to prevent the generative model to copy and loop the conditioning audio. We thus need to introduce and tune information bottlenecks in our conditioning module. Our contributions are the following:

1) We adapt the textual inversion method of [9] to a pretrained text-to-music MusicGen model. This allows to perform audio conditioning for music generation without training a model from scratch.

2) We present our style conditioner method which is based on a frozen audio feature extractor (Encodec [10], MERT [11] or MusicFM [12]) followed by a transformer encoder [13], Residual Vector Quantizer (RVQ) [14] and temporal downsampling. The number of residual streams used by RVQ is adjustable at inference time which gives the user the ability to change the strength of the style conditioning. To our knowledge, we are the first to explore

this approach for music generation.

3) Since the model is trained with both textual and audio conditioning inputs, we can combine both to generate music. However, audio contains much more information, so that text is ignored by the model at inference. We propose to balance them with a new double classifier free guidance [15] which is a general method for merging conditions with various degrees of information.

4) We introduce novel objective metrics for style conditioning, based on nearest neighbors search in the latent space, validated with human evaluations.

We compare our method to baselines which are: a MusicGen trained with CLAP embeddings [16] as conditioning, a text-to-music MusicGen used with text prompts, and a MusicGen model without conditioning used in continuation mode. We perform as well some ablation studies in order to justify the architecture of our style encoder. Based on results, we show the practicality of our methods and the musical quality of the generated music.

2. RELATED WORK

2.1 Generative models for music

Music generation models can be categorized into two types: autoregressive models and non autoregressive ones. **Autoregressive** ones are motivated by the successful work done in natural language modeling. Recent successful models use a compression model taking the form of a multi stream quantized autoencoder [10, 14] in order to convert audio into K parallel discrete streams of tokens. The K streams are obtained by performing Residual Vector Quantization (RVQ) [14] on the latent space of an autoencoder, making the first stream contain coarse information and following ones refine the approximation of the latent space. Then, an autoregressive transformer [13] is used to model these audio tokens. MusicLM [4] and MusicGen [5] are built on this principle. MusicLM uses a multi-stage approach with different models to predict the K streams, while MusicGen models them in parallel using a delay pattern [5, 17].

Non-autoregressive models such as AudioLDM2 [18], MusicLDM [19], and Stable Audio [6], are latent diffusion models operating in the latent space of a continuous variational autoencoder. Some other models use cascaded diffusion such as Noise2Music [20] to progressively increase the sampling rate of the audio. Moûsai [21] uses a first diffusion model to compress the music and a second one to generate music from this representation and textual descriptions. MusTango [7] uses a latent diffusion model conditioned on textual description, chord, beat, tempo and key. Jen-1 [22] combines a diffusion model and a masked autoencoder trained with multi-tasks objectives. It can perform music generation, continuation and inpainting. A second version [23] uses source separation [24] over their dataset to allow the user to generate and edit music stem by stem. VampNet [25] is a masked modeling approach to music synthesis that uses masking at training and inference time in order to generate discrete audio tokens. MAGNeT [26] is based on the same masking principle. It can also combine autoregressive and masking to reach the same quality as the autoregressive baseline (MusicGen) but with a 7x faster inference. In MeLoDy [27], a language model is used to model coarse semantic tokens and a dual path diffusion model is then used for acoustic modeling. The authors claim faster than real time generation.

2.2 Jointly trained conditioners for music generative models

Regarding the conditioning, most of the models focused on text-to-music [4, 5, 19–22]. Since pairs of text-music data are rare, most models use a pre-trained contrastive text-music model such as CLAP [16] or MuLan [28], to condition their text-to-music models. Then, massive amount of non-annotated audio data can be used at training time and text is used at inference time. However, these text-to-music models never exploit the fact that audio can be used as conditioning. For other types of conditioning, MusTango [7] is trained with text, beat tempo, key and chords as conditioning, StableAudio [6] takes timing embeddings to control the length and structure of the generated music. Some models generate stems while being conditioned on other stems. For instance, SingSong [29] generates musical accompaniments from singing and Jen-1 Composer [23] handles multi-track music generation on 4 different stems (bass, drums, instrument and melody). MusicGen [5] and Music ControlNet [30] can handle melody as conditioning and the latter can also use dynamics and rhythm. Both papers use chromagrams extraction for melody conditioning.

2.3 Conditioning a pretrained generative model

With finetuning: In Coco-Mulla [8], the authors use parameter-efficient fine-tuning (PEFT) to specialize a text-to-music MusicGen model on chords and rhythm. They finetune on a number of parameter that is 4% the amount of parameters of the original network with only 300 songs. Music ControlNet [30] is a finetuned text-to-music diffusion model that operates in the spectral domain. The finetuning strategy comes from the text-to-image method ControlNet [31] and allows to handle melody, dynamics and rhythm conditioning. The pixel-level control that allows ControlNet on images gives a pixel-level control on the mel-spectrogram.

Without finetuning: In [32], the authors use AudioLDM [18] as a backbone to perform textual inversion [9]. For each textual inversion they use a group of 5 excerpts of 10 seconds. They also try an experiment where they optimize the pseudoword S^* as well as the diffusion neural network which gives them better results. In [33], the authors use a diffusion model trained on musical data with no conditioning and perform various interactive tasks at inference which are infilling, continuation, transition (smooth a transition between two songs) and guidance. The one that is the most similar to our audio conditioning is the guidance where the diffusion model is guided by the PaSST classifier [34] embedding of an audio prompt. However the model only generates 5 seconds excerpts of music. Some

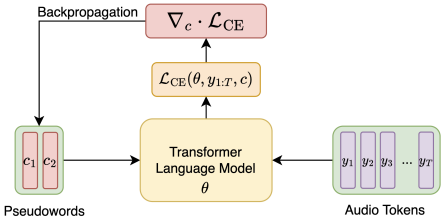

Figure 1. An overview of the Texual Inversion method based on the pretrained text-to-music MusicGen

other papers involve new control with no finetuning such as in [35] or DITTO [36] where the authors use a pre-trained text-to-music diffusion model and control its inference by optimizing the initial noise latent. In SMITIN [37], the authors control a pretrained MusicGen model by steering the attention heads in the direction that maximizes the probability of some features.

3. TEXTUAL INVERSION METHOD

We first present our textual inversion method in the case of autoregressive modeling (see Fig. 1). It is based on previous work in the image domain [9] with diffusion models.

Autoregressive modeling aims to estimate the conditional distribution of the next token y_t given the preceding tokens $y_{<t}$ and a conditioning context c, such as a textual embedding. In the framework of transformer decoder neural networks parameterized by θ, denoted as p_θ, this conditional distribution is typically modeled as a product of individual probabilities:

$$p_\theta(y_{1:T}|c) = \prod_{t=1}^{T} p_\theta(y_t|y_{<t}, c) \tag{1}$$

Here, $y_{1:T}$ represents the sequence of tokens, and $p_\theta(y_t|y_{<t}, c)$ denotes the probability of observing token y_t given the preceding tokens and the conditioning context. During training, with a given sequence $y_{1:T}$ and its associated textual description c, we compute the cross-entropy loss:

$$\mathcal{L}_{\text{CE}}(\theta, y_{1:T}, c) = -\sum_{t=1}^{T} \log p_\theta(y_t|y_{<t}, c) \tag{2}$$

It is minimized by taking a gradient descent step on $\nabla_\theta \mathcal{L}_{\text{CE}}(\theta, y_{1:T}, c)$. This loss quantifies the dissimilarity between the predicted conditional distribution and the true distribution of the next token, serving as the optimization objective for training autoregressive models.

For the textual inversion method, we take a pretrained text-to-music MusicGen for the transformer decoder. We initialize the textual embedding (for instance with the textual embedding of the word "music") c. Given a song Y, we cut it into random chunks $\{y^i_{1:T}\}_i$ and optimize the textual embedding c by taking successive gradient steps on $\nabla_c \mathcal{L}_{\text{CE}}(\theta, y^i_{1:T}, c)$. After a few hundreds iterations the learnt c is fed into the text-to-music MusicGen model to generate a song in the style of Y.

4. STYLE CONDITIONING METHOD

4.1 General Architecture

The general architecture, depicted on the left of Fig. 2, is based on the text-to-music model MusicGen [5] with the addition of a style conditioner that is jointly trained with the language model. At train time, a 30 seconds music excerpt paired with a textual description is input to the model. The textual description is fed into a frozen T5 tokenizer and transformer encoder [38]. The style encoder takes a random subsample (between 1.5 and 4.5 seconds) of the input audio and encodes it. The text and style latent representations are both projected with linear layers to have the same dimension as the transformer language model, and provided as prefix to the sequence to model.

The input audio is encoded by a pretrained EnCodec [10] model and the language model is trained in a autoregressive manner with a cross-entropy loss. In addition, the tokens that correspond to the random subsample fed into the style encoder are masked in the loss, as we noticed this reduces the tendency of the model to just copy the style audio input. At inference time, we can use text or/and a short excerpt of music as a conditioning to generate music.

4.2 Architecture of the Style Conditioner

Our style conditioner is designed with bottlenecks (RVQ [14] and downsampling) to prevent transmitting all the information of the conditioning audio excerpt to the model. Without these bottlenecks, the generative models retrieves easily the excerpt and copies it (see the ablation study in Sec. 5.5). The style conditioner depicted on the right of Fig. 2 takes an audio input of length 1.5 to 4.5 seconds, passes it through a frozen feature extractor followed by a trainable transformer encoder and a residual vector quantization (RVQ) module with 6 codebooks. After quantization, we downsample on the temporal axis to obtain a conditioning with a 5Hz frame rate which gives a similar length as a text description (8 to 25 tokens). Finally a linear layer outputs the same dimension as the language model.

The candidates for the audio encoder are a Encodec followed by trainable embeddings for each codebook that are summed, a transformer based music foundation model from [12] (we now name it MusicFM for the rest of the paper) where the authors claim state of the art on several downstream tasks specific to music information retrieval and a MERT model [11], a transformer based music model trained in a self-supervised manner. The first one has a frame rate of 50Hz and 60M parameters, the second one has a frame rate of 25Hz and 620M parameters and the third one has a frame rate of 75Hz and 95M parameters

At training time, we use dropout on the conditioning, keeping both conditions 25% of time, one of the two conditions 25% of time for each (no text or no style) or no condition 25% of time. There is also a dropout on the number of the codebooks used by the RVQ of the style conditioner: at each step of the training, the number of used codebooks is uniformly sampled between 1 and 6. Then, at inference time, we can control the bottleneck of the style conditioner.

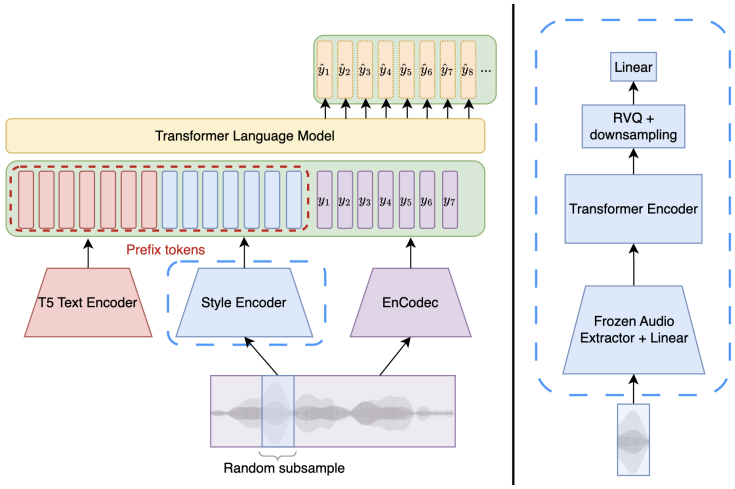

Figure 2. An overview of the general architecture. Text conditioning and style conditioning are provided to the model as a prefix. On the right we present the style conditioner.

Setting the number of codebooks to 1 gives more flexibility to the generative model whereas using 6 levels of quantization constraints it more. In practice, this means that music generated with 6 streams of quantization will sound more similar to the input condition compared to music generated with 1 stream of quantization.

4.3 Double Classifier Free Guidance

When doing next token prediction, let's denote $l_{\text{style, text}}$ the logits of the model conditioned on style and textual description. Classifier free guidance [15] consists of pushing the logits in the direction predicted with the conditioning, to increase its importance:

$$l_{\text{CFG}} = l_{\emptyset} + \alpha(l_{\text{style, text}} - l_{\emptyset}), \text{with } \alpha > 1, \quad (3)$$

typically, $\alpha = 3$ is used in previous work [5].

When generating music with a textual description that contradicts the audio of the style conditioning, we observe that the description is ignored by the model. This is explained by the fact that audio is more informative conditioning compared with the text, so that the model weights it more during training. To counteract this effect, we introduce a *double classifier free guidance* in which we iterate the CFG formula: we first push from style only to style and text and we then push these logits a second time from no conditioning.

$$l_{\text{double CFG}} = l_{\emptyset} + \alpha[l_{style} + \beta(l_{text,style} - l_{style}) - l_{\emptyset}] \quad (4)$$

We retrieve the simple CFG with $\beta = 1$. For $\beta > 1$, we boost the importance of the text conditioning (see Sec. 5.6).

4.4 Objective Metrics

The difficulty with generating samples in the same style of a song is that we want to generate something that is similar enough but not too close. This is something that can be subjectively evaluated. For easing the comparison of various approaches and hyper parameters, we also introduce a novel set of objective metrics.

Nearest Neighbours in Common: Let's note $x_C \in \mathbb{R}^{D \times T}$ ($D = 1$ for mono music) the audio that we input in the style conditioner and $x_G \in \mathbb{R}^{D \times T'}$ the generated sequence. We use an encoder $E : \mathbb{R}^{D \times T} \to \mathbb{R}^N$ which outputs a single vector whatever the input length T is. In practice, this is done by taking a MusicFM model and averaging on the time dimension. Then, for each song of our valid and test sets, we cut it into chunks of 30 seconds and store the embeddings $\{E_{i,j}\}$, i being the index of the song and j the chunk number. For $E_C = E(x_C)$, we compute the cosine similarities $\cos(E_C, E_{i,j}), \forall i, j$ and the set of its K nearest neighbors: $\{i_1^C, ...i_K^C\}$. We do the same for $E_G = E(x_G)$ and obtain a set of K values $\{i_1^G, ...i_K^G\}$. We then have found the nearest songs in the dataset. We define our metric $\text{KNN}_{\text{common}}(x_C, x_G)$ for a song x_G that has been generated after being conditioned by x_C:

$$\text{KNN}_{\text{common}}(x_C, x_G) = \frac{|\{i_1^C, ...i_K^C\} \cap \{i_1^G, ...i_K^G\}|}{K} \in [0, 1]. \quad (5)$$

The intuition behind this metric is that a model performs well at recreating a song in the style of another if the generated song and its conditioning audio have embeddings that are close enough to share neighbors in the dataset. However, if a model copies the conditioning (i.e. $x_G \approx x_C$) the metric will tend to 1, we thus need a second metric to avoid x_G and x_C being too similar.

G is the Nearest Neighbor of C: We want E_G and E_C to be close while being different. One way to be sure that the corresponding audios are not too similar is to check that if we add E_G to the set of embeddings $\{E_{i,j}\}$, E_G is not the nearest neighbor of E_C. Assuring that another song from the dataset is closer to the conditioning means that the model is creative enough and not just copying its input. Formally, denoting $\{E_\cup\} = \{E_{i,j}\} \cup \{E_G\}$, we define

$$\text{KNN}_{\text{overfit}}(x_C, x_G) = \begin{cases} 1 \text{ if } \underset{E \in \{E_\cup\}}{\text{argmax}} [\cos(E_C, E)] = E_G \\ 0 \text{ otherwise.} \end{cases} \quad (6)$$

Model	$FAD_{vgg}\downarrow$	KL $\downarrow$	CLAP $\uparrow$	$KNN_{common}\uparrow$	$KNN_{overfit}\downarrow$	OVL $\uparrow$	SIM $\uparrow$	VAR $\downarrow$
Textual Inversion	6.07	0.55	0.20	0.20	0.14	78.11 ± 0.93	61.78 ± 1.06	69.53 ± 1.44
MusicGen Continuation	1.22	0.51	0.30	0.26	0.17	83.95 ± 0.83	73.38 ± 0.97	77.24 ± 1.29
MusicGen w. audio CLAP	0.96	**0.43**	**0.31**	0.09	0.02	84.76 ± 0.93	62.37 ± 1.04	68.58 ± 1.42
Our Model w. EnCodec, 2 RVQ	**0.85**	0.49	0.29	0.23	0.12	83.41 ± 1.04	72.16 ± 0.93	72.39 ± 1.33

Table 1. Comparison with baselines. The KNN_* metrics, introduced in Sec. 4.4, measure how close the generation is from the style condition, yet different from the matching ground truth. Those are completed with the subjective evaluations from Sec. 4.5. While using MusicGen for continuation matches well to the style audio, it has limited variation. Using a CLAP audio encoder as conditioning does the opposite, while using our style encoder gets the right balance between the two.

For our evaluations, we take 1000 samples of 3 seconds x_C from our test set, generate the corresponding x_G and average the two KNN metrics. Intuitively, the two metrics are positively correlated, but for a similar value for KNN_{common} we will favor the model that minimizes $KNN_{overfit}$.

Other Objective Metrics To evaluate the quality of the generated music, we also use the official implementation of the Fréchet Audio Distance defined in [39] that uses a VGGish model, the KL-divergence based metric introduced in [5] that computes the KL-divergence on the probabilities of the labels of a pretrained audio classifier between the conditioning and the generated music. We noticed that a high FAD (> 2) often indicates a poor quality of the generated samples. The CLAP score [5, 16] computes the cosine similarity between the description and the audio embeddings obtained with the CLAP model. A higher score indicates that the generated audio aligns well with the textual description of the conditioning audio.

4.5 Human studies metrics

We follow a similar protocol as in [5] for the human studies. We ask human raters to evaluate three different aspects of the generated audio: (1) How would you rate the overall quality of this excerpt [OVL]? (2) Without considering audio quality, how similar are these two excerpts in terms of style [SIM]? (3) Without considering audio quality, how likely do you think these two excerpts are from the same song [VAR]?

We believe that the SIM and VAR scores are the subjective versions of KNN_{common} and $KNN_{overfit}$.

5. EXPERIMENTAL RESULTS

5.1 Hyperparameters for the textual inversion

For the textual inversion method we test different parameters sets and retain these ones: we use a 12 tokens sentence for initialization, a batch size of 8 with 5 seconds segments randomly sampled from a 30 second excerpt with 200 optimization steps, a learning rate of 0.025 with a vanilla Adam optimizer. Finally the main issue that we encounter with this method is its instability. It is hard to find a set of hyperparameters that works well for any song. Some songs seem to be easier to invert for different sets of hyperparameters. For some song, we never achieve to obtain hearable music as the result suffers from glitches, and tempo instabilities. Finally, it seems beneficial to augment the length of the text embedding, as well as performing the inversion over chunks taken from a 30 seconds excerpt rather than the entire song. The results are shown in Tab. 1.

5.2 Hyperparameters for the style conditioner

All the models that we train are medium size (1.5B parameters) MusicGen models built on top of the 4 stream 32kHz music version of EnCodec [10]. All models are trained for 400K steps on 64 V100 GPUs with the AdamW optimizer using $\beta_1 = 0.9$, $\beta_2 = 0.95$, a batch size of 192, and music sequences of 30 seconds. For the style conditioner, excerpts between 1.5 and 4.5 seconds are subsampled from the original sequence, the transformer encoder has 8 layers, 8 heads, a dimension of 512 and is non-causal, the residual vector quantizer has a codebook size of 1024, 6 streams and a variable number of streams is sampled at each training step, hence allowing the language model to train on all the levels of quantization. The style tokens are downsampled to 5Hz. All our evaluations are done on 1000 samples of the test set. Similarly to the MusicGen Melody model, both the textual description and the style condition are provided as prefix to the language model.

5.3 Datasets

We use 20K hours of licensed music as in [5]. The training dataset is composed of 25K and 365K songs from the ShutterStock and Pond5 music data collections, as well as 10k tracks of an internal dataset. Each song comes with textual description, and is downsampled to 32kHz mono.

5.4 Comparison with baselines and model selection

Apart from the closed-source model udio [40], there is no other audio conditioned music generative model. We use as a baseline a MusicGen model in the continuation setting: given 3 seconds of music, we ask MusicGen to continue the music with no textual prompt. For the second one we train a MusicGen model with a pretrained CLAP audio encoder [16] as conditioning, also taking 3 seconds of audio as input. In Tab. 1, we compare these two baselines with our model with the EnCodec feature extractor for the style conditioner, a quantization level of 2 and with a textual inversion model. We notice that the FAD correlates well with the quality metric (OVL) since the textual inversion model has the worst OVL and FAD scores. Thus excluding this approach, we observe that the KNN_{common} and the SIM metrics ranks the models in the same orders as well as the $KNN_{overfit}$ and VAR metrics.

Feat. Ext.	Quant.	$FAD_{vgg} \downarrow$	KL $\downarrow$	CLAP $\uparrow$	$KNN_{common} \uparrow$	$KNN_{overfit} \downarrow$	OVL $\uparrow$	SIM $\uparrow$	VAR $\downarrow$
MERT	1	0.78	0.50	0.29	0.19	**0.06**	84.07 ± 0.93	70.27 ± 1.22	$\mathbf{69.69 \pm 1.31}$
MERT	2	0.75	0.47	0.30	0.24	0.13	84.14 ± 0.96	72.53 ± 1.05	72.81 ± 1.21
MERT	4	0.75	**0.45**	**0.31**	**0.29**	0.18	$\mathbf{84.32 \pm 1.04}$	$\mathbf{74.15 \pm 0.96}$	75.12 ± 1.35
EnCodec	2	0.85	0.49	0.29	0.23	0.12	84.02 ± 0.89	72.69 ± 0.91	72.47 ± 1.28
MusicFM	2	**0.70**	**0.45**	**0.31**	0.28	0.16	84.45 ± 1.09	73.01 ± 0.95	74.01 ± 1.36

Table 2. Comparison between the 3 feature extractors. The human studies correlate well with the KNN metrics. As expected, using coarser quantization of the style features leads to more variations in the generated audio. Self-supervised encoder like MERT and MusicFM outperforms low level acoustic models like EnCodec.

Model	$FAD_{vgg} \downarrow$	KL $\downarrow$	CLAP $\uparrow$	$KNN_{common} \uparrow$	$KNN_{overfit} \downarrow$
Our Model	**0.75**	**0.45**	**0.31**	0.29	0.18
Smaller Transformer	0.98	0.48	0.29	0.24	0.13
No Transformer	2.92	0.96	0.13	0.01	0.0
No Masking of the loss	1.11	0.53	0.29	0.22	0.30

Table 3. Ablation Study on our model with a MERT feature extractor with 4 quantization streams.

Regarding the baselines, the textual inversion method provides results of poor quality (FAD). The continuation method provides music that has a high similarity to the conditioning (high KNN_{common} and SIM) but that is too similar to it (high $KNN_{overfit}$ and VAR). However, the CLAP conditioning captures a more vague style of the conditioning and generates music that is too far from it (low KNN_{common}, $KNN_{overfit}$, SIM and VAR). Our model with the EnCodec feature extractor and 2 levels of quantization strikes the right balance between these two baselines.

In order to strengthen our claim that our KNN metrics correlates well with human perception of closeness between musical excerpts, we showcase a second study in Tab. 2. In this study we compare the metrics of the MERT feature extractor with 3 quantization levels 1, 2, 4 (we recall that the models can go up to 6) as well as the EnCodec and MusicFM feature extractors with a quantization level of 2. All models generate music of similar quality (FAD and OVL). We notice that when the bottleneck is larger (i.e. when the quantization level is higher), the KNN_{common} augments. This follows the intuition that if the conditioner transmits more information to the language model, the generated music will be closer to the input condition. The models follows similar orders for KNN_{common} and SIM as well as for $KNN_{overfit}$ and VAR.

5.5 Ablation Study

We perform an ablation study in Tab. 3 on the components of the style conditioner with MERT as a feature extractor, and 4 RVQ streams. When reducing the size of the transformer encoder from 8 layers and 512 dimensions to 4 layers and 256 dimensions, the quality of the generated audio is worse. When removing the transformer encoder, the model generates audio that is far from music (high FAD). When we don't mask the music that is input to the style conditioner in the cross-entropy loss at training time, the audio quality is slightly worse and the model generates music that is too close to the conditioning and tends to loop. The very high $KNN_{overfit}$ indicates it since for a KNN_{common} lower than the best model the $KNN_{overfit}$ is twice its value.

Type	α	β	$FAD_{vgg} \downarrow$	CLAP $\uparrow$	$KNN_{common} \uparrow$
No CFG	✗	✗	1.54	0.25	0.088
simple	3	✗	0.92	0.28	0.162
double	3	3	0.80	0.35	0.123
double	3	4	0.78	0.37	0.104
double	3	5	0.84	0.37	0.095
double	3	6	0.97	0.38	0.081

Table 4. Classifier Free Guidance parameters tuning. Larger β from (4) leads to increasing the importance of the text conditioning (given by the CLAP score), and decreasing the similarity to the style audio, given by KNN_{common}.

5.6 Tuning the Classifier Free Guidance

When style and text conditioning are both used and are not consistent, it is necessary to use double CFG instead of simple CFG so that the text is not ignored. To tune the parameters α, β of the double classifier free guidance given by (4), we rely on the following protocol. For 1000 samples of our test set, we randomly shuffle text descriptions and generate music while conditioning both on text and music. We track the FAD [39], the KNN_{common} and the CLAP score. In Tab 4 we observe the intuitive fact that the KNN_{common} and CLAP score are negatively correlated: if the balancing favors the text condition the CLAP score is higher, if it favors the audio condition the KNN_{common} is higher. The double CFG thus works as expected.

6. CONCLUSION

In this paper we introduced style conditioning for language model based music generative models: given a few seconds of a musical excerpt, one can generate music in the same style using our proposed audio encoder with an information bottleneck. We introduced new metrics to assess the equilibrium between generating music that maintains a similar style to the condition while also being different. We validated those with human studies. Finally, we can also mix this style conditioning with inconsistent textual description and balance them thanks to a new double classifier free guidance method. This method could be applied in other generative models with multiple conditions.

Ethical statement: Improving music generation brings ethical challenges. Through carefully chosen bottlenecks in our style extractor (RVQ, downsampling) we aim for the right balance between increasing the model controllability and possible creative use while ensuring the model does not copy existing works, and provided new metrics to measure this. Finally, we only used music we licensed.

7. REFERENCES

[1] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," in *ICLR*, 2019.

[2] S. Rouard and G. Hadjeres, "Crash: Raw audio score-based generative modeling for controllable high-resolution drum sound synthesis," in *ISMIR*, 2021.

[3] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," in *ISMIR*, 2017.

[4] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, M. Sharifi, N. Zeghidour, and C. Frank, "Musiclm: Generating music from text," *ArXiv*, 2023.

[5] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," in *Neurips*, 2023.

[6] Z. Evans, C. Carr, J. Taylor, S. H. Hawley, and J. Pons, "Fast timing-conditioned latent audio diffusion," *ArXiv*, 2024.

[7] J. Melechovsky, Z. Guo, D. Ghosal, N. Majumder, D. Herremans, and S. Poria, "Mustango: Toward controllable text-to-music generation," *ArXiv*, 2023.

[8] L. Lin, G. Xia, J. Jiang, and Y. Zhang, "Content-based controls for music large language modeling," *ArXiv*, 2023.

[9] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, "An image is worth one word: Personalizing text-to-image generation using textual inversion," in *ICLR*, 2023.

[10] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *TMLR*, 2022.

[11] Y. Li, R. Yuan, G. Zhang, Y. Ma, X. Chen, H. Yin, C. Lin, A. Ragni, E. Benetos, N. Gyenge, R. Dannenberg, R. Liu, W. Chen, G. Xia, Y. Shi, W. Huang, Y. Guo, and J. Fu, "Mert: Acoustic music understanding model with large-scale self-supervised training," *ArXiv*, 2023.

[12] M. Won, Y.-N. Hung, and D. Le, "A foundation model for music informatics," *ICASSP 24*, 2024.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[14] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, 2022.

[15] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021.

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

[17] E. Kharitonov, A. Lee, A. Polyak, Y. Adi, J. Copet, K. Lakhotia, T. A. Nguyen, M. Riviere, A. Mohamed, E. Dupoux, and W.-N. Hsu, "Text-free prosody-aware generative spoken language modeling," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 8666–8681.

[18] H. Liu, Q. Tian, Y. Yuan, X. Liu, X. Mei, Q. Kong, Y. Wang, W. Wang, Y. Wang, and M. . Plumbley, "Audioldm 2: Learning holistic audio generation with self-supervised pretraining," *ArXiv*, 2023.

[19] K. Chen, Y. Wu, H. Liu, M. Nezhurina, T. Berg-Kirkpatrick, and S. Dubnov, "Musicldm: Enhancing novelty in text-to-music generation using beat-synchronous mixup strategies," *ArXiv*, 2023.

[20] Q. Huang, D. S. Park, T. Wang, T. I. Denk, A. Ly, N. Chen, Z. Zhang, Z. Zhang, J. Yu, C. Frank, J. Engel, Q. V. Le, W. Chan, Z. Chen, and W. Han, "Noise2music: Text-conditioned music generation with diffusion models," *ArXiv*, 2023.

[21] F. Schneider, O. Kamal, Z. Jin, and B. Schölkopf, "Moûsai: Text-to-music generation with long-context latent diffusion," *ArXiv*, 2023.

[22] P. Li, B. Chen, Y. Yao, Y. Wang, A. Wang, and A. Wang, "Jen-1: Text-guided universal music generation with omnidirectional diffusion models," *ArXiv*, 2023.

[23] Y. Yao, P. Li, B. Chen, and A. Wang, "Jen-1 composer: A unified framework for high-fidelity multi-track music generation," *ArXiv*, 2023.

[24] S. Rouard, F. Massa, and A. Défossez, "Hybrid transformers for music source separation," in *ICASSP 23*, 2023.

[25] H. F. Garcia, P. Seetharaman, R. Kumar, and B. Pardo, "Vampnet: Music generation via masked acoustic token modeling," *ArXiv*, 2023.

[26] A. Ziv, I. Gat, G. L. Lan, T. Remez, F. Kreuk, A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "Masked audio generation using a single non-autoregressive transformer," in *ICLR*, 2024.

[27] M. W. Y. Lam, Q. Tian, T. Li, Z. Yin, S. Feng, M. Tu, Y. Ji, R. Xia, M. Ma, X. Song, J. Chen, Y. Wang, and Y. Wang, "Efficient neural music generation," in *Neurips*, 2023.

[28] Q. Huang, A. Jansen, J. Lee, R. Ganti, J. Y. Li, and D. P. W. Ellis, "Mulan: A joint embedding of music audio and natural language," in *ISMIR*, 2022.

[29] C. Donahue, A. Caillon, A. Roberts, E. Manilow, P. Esling, A. Agostinelli, M. Verzetti, I. Simon, O. Pietquin, N. Zeghidour, and J. Engel, "Singsong: Generating musical accompaniments from singing," *ArXiv*, 2023.

[30] S.-L. Wu, C. Donahue, S. Watanabe, and N. J. Bryan, "Music controlnet: Multiple time-varying controls for music generation," *ArXiv*, 2023.

[31] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *ICCV*, 2023.

[32] M. Plitsis, T. Kouzelis, G. Paraskevopoulos, V. Katsouros, and Y. Panagakis, "Investigating personalization methods in text to music generation," *ArXiv*, 2023.

[33] M. Levy, B. D. Giorgi, F. Weers, A. Katharopoulos, and T. Nickson, "Controllable music production with diffusion models and guidance gradients," *ArXiv*, 2023.

[34] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," in *Interspeech 2022*. ISCA, Sep. 2022.

[35] H. Manor and T. Michaeli, "Zero-shot unsupervised and text-based audio editing using ddpm inversion," in *ICML*, 2024.

[36] Z. Novack, J. McAuley, T. Berg-Kirkpatrick, and N. J. Bryan, "Ditto: Diffusion inference-time t-optimization for music generation," *ArXiv*, 2024.

[37] J. Koo, G. Wichern, F. G. Germain, S. Khurana, and J. L. Roux, "Smitin: Self-monitored inference-time intervention for generative music transformers," 2024. [Online]. Available: https://arxiv.org/abs/2404.02252

[38] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[39] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A metric for evaluating music enhancement algorithms," *ArXiv*, 2019.

[40] Udio. [Online]. Available: https://www.udio.com