Latent audio diffusion for music generation with expressive control

Alfred Pichard **Hugo Audas** **Paul Triana** **Tom Oviste**

Abstract

Recent advances in generative deep learning models provide exciting new tools for music generation. In particular, the conditioning capabilities of diffusion models offer interesting possibilities to add expressive control to the generation process. In this paper, we apply the novel diffusion method Iterative α-(de)Blending [HBC23], which simplifies the usual formalism of stochastic diffusion models to a deterministic one, to generate rhythmic audio loops from pure noise. We use a high-fidelity neural autoencoder [DCSA20] to generate latent codes of a compressed audio representation. Conditioning is applied using either beats-per-minute information or high-level audio concepts [WCZ$^+$23] and reinforced using classifier-free guidance. The latent codes are then inverted back to a waveform with the decoder. Finally, we assess the quality of our method on a large dataset of minimal electronic music.

1 Introduction

While numerous state-of-the-art generative deep learning networks such as Generative Adversarial Networks, Variational Auto Encoders [SYL15] or Normalizing Flows [RM15] have successfully been used as prior models for generating latent codes, these attempts often suffered from unreliability and unsteadiness which represent costs for training and generating data. However, diffusion models have garnered significant attention for both their accuracy and training stability, but also for their conditioning capabilities, offering unique possibilities to introduce expressive control into the generation process. Hence, these distinctive features present a compelling case for their utilization in music generation. This research delves into the exploration of how diffusion models can serve as effective prior models for generating latent codes within an audio compression framework. Harnessing the expressive potential of such models, we also aim to implement high-level controls over the generation process.

Moreover, as it introduces a simpler formulation and a deterministic version of diffusion models, we will rely on the recent results of Iterative α-(de)Blending (IADB) [HBC23] and U-Net [RB15] convolution network to generate our data. Our approach also involves employing the high-fidelity neural autoencoder EnCodec [DCSA20] to generate the latent codes representing compressed audio representation. The latent codes are subsequently inverted to reconstruct a waveform using the decoder. As for conditioning, we use both continuous audio descriptors such as beats-per-minute information, or high-level audio concepts extracted from our dataset via the results of CLAP[WCZ$^+$23] to train our network and infer our results. We provide some additional reinforcement in our conditioning through classifier-free guidance [HS22].

Finally, we use a large dataset of minimal house and techno music for training, focusing on the repetitive and percussive elements that partially define this subgenre. This strategic approach aims to enhance our model's ability to capture and reproduce those distinctive features, resulting in more genre-specific generative outputs.

Preprint. Under review.

2 State of the art of diffusion models

2.1 Diffusion probabilistic models

The original formulation of diffusion probabilistic models was inspired by considerations from non equilibrium thermodynamics [SDWMG15]. By gradually converting one distribution $x_0 \sim q_0$ into Gaussian noise $x_T \sim \mathcal{N}(0,1)$, defining the series of latent variables $\mathbf{x}_1, \cdots, \mathbf{x}_T$ with the diffusion or forward process $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, denoising $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ can be learned, and help generate high-quality waveforms from pure noise.

Applying the diffusion process as a fixed Markov chain to add Gaussian noise to the data according to a variance schedule $\beta_1, \cdots, \beta_T$ yields:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

The complete diffusion process is defined as the joint distribution $q(\mathbf{x}_{0:T})$:

$$q(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

Each denoising step is a Markov chain with learned Gaussian parameters starting from pure noise $p(\mathbf{x}_T) \sim \mathcal{N}(0,1)$

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t))$$

The complete denoising process is defined as the joint distribution $p_\theta(\mathbf{x}_{0:T})$:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

The model's training loss $\mathcal{L}$ is eventually defined by applying Jensen's inequality to the negative log-likelihood:

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}\right] = \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})}\right] = \mathcal{L}$$

Training is then be performed by optimizing random terms of $\mathcal{L}$ with stochastic gradient descent

2.2 A simpler deterministic diffusion model: iterative α-(de)blending

As the formulation of diffusion models involves advanced theoretical knowledge, and considering the proven efficiency of such a deep learning approach, we were pleased that the authors of IADB [HBC23] successfully landed on a minimalist version of diffusion models. In doing so they were able to produce a deterministic mapping between the two distributions p_0 and p_1 using samples following these distributions $x_0 \sim p_0$ and $x_1 \sim p_1$ and a blended sample $x_\alpha \sim p_\alpha$ such that $x_\alpha = (1-\alpha)x_0 + \alpha x_1$ with $\alpha \in [0,1]$.

α-(de)blending consists in sampling posteriors $(x_0, x_1)_{(x_\alpha, \alpha)} \sim (p_0 \times p_1)_{(x_\alpha, \alpha)}$ knowing x_α and α.

Hence, IADB relies on setting a number of iterations T and a set of $\alpha_t = \frac{t}{T}$ with $t \in [0,T]$ evenly distributed to create a sequence of $(x_{\alpha_t} \sim p_{\alpha_t}, t \in [0,T])$ starting with a random sample $x_{\alpha_0} = x_0 \sim p_0$ and ending with a random sample $x_{\alpha_T} = x_T \sim p_T$. Each sample x_{α_t} is then deblended to find posterior samples. These posteriors are blended once more with α_{t+1} to find a new sample $x_{\alpha_{t+1}} \sim p_{\alpha_{t+1}}$. These blending and deblending steps are repeated iteratively.

IADB can eventually be learned to predict the average posterior samples $\bar{x}_0$ and $\bar{x}_1$. In practice, however, the authors learn the prediction of $\bar{x}_0 - \bar{x}_1$ as it provides a more stable implementation.

Once D_θ is trained it is possible to generate a sequence of samples ($x_{\alpha_t} \sim p_{\alpha_t}, t \in [0, T]$) by gradually de-noising a random sample from the first distribution p_0 according to a given schedule of $t \in [0, T]$ by applying the recurrent formula:

$$x_{\alpha_{t+1}} = x_{\alpha_t} + (\alpha_{t+1} - \alpha_t) D_\theta(x_{\alpha_t}, \alpha_t)$$

2.3 Conditioning: Classifier Guidance and Classifier-Free Guidance

Conditioning refers to the methods that aim to add *control* to generative models. For example, in image generation, the conditional diffusion process generates an image that follows the description of a text prompt. In our case, we want the diffused waveform to follow the rhythm of given beats, or to sound similar to another given audio waveform. Then, as explained later, we apply conditioning by concatenating new information to the latent code to be diffused – either a saw-like signal that represents BPM, or a CLAP signal that represents high-level audio concepts – and by training the diffusion model accordingly.

Classifier guidance is a method that steers diffusion sampling towards the direction that maximizes the probability that the final sample gets classified as a particular class [DN21]. Indeed: firstly, at a given denoising step, we can take the update direction obtained from the diffusion model $\nabla_{x_t} \log p_t(x_t)$. Now, suppose that we possess a classifier model that predicts $p(y|\mathbf{x_t})$, where y represents an arbitrary input feature – for example, y could be a class label or, in our case, a signal representing the BPM of a latent code. Using this classifier, we can determine the direction in input space that maximizes the log-likelihood of the conditioning signal by computing the gradient with respect to x_t: $\nabla_{x_t} \log p(y|x_t)$. Scaling this direction and adding the unconditional update direction, we obtain $\nabla_{x_t} \log p_t(x_t) + \gamma \nabla_{x_t} \log p(y|x_t)$, where scaling factor γ is called *temperature*. This expression corresponds to the gradient of $p_t(x_t) \cdot p(y|x_t)^\gamma$: in other words, we weight the distribution of the model towards the direction where the classifier assigns the desired label. The guided model then becomes much more likely to generate samples that align with the desired label [Die23]. However, the main drawback of classifier guidance is that it requires an external classifier, which needs to be trained separately. Moreover, this classifier has to be noise-robust, since it is applied to noisy versions of the input.

To fix this issue, classifier-free guidance is a modern variant of classifier guidance that does not require an external classifier model [HS22]. Hence, the main idea consists of training the one diffusion model as both a conditional generative model by computing $\nabla_{x_t} \log p_t(x_t)$, and an unconditional generative model by computing $\nabla_{x_t} \log p_t(x_t|y)$. Using Bayes' rule, we can then reformulate the update direction and obtain:

$$\nabla_{x_t} \log p_t(x_t) + \gamma \nabla_{x_t} \log p(y|x_t) = (1 - \gamma) \nabla_{x_t} \log p_t(x_t) + \gamma \nabla_{x_t} \log p_t(x_t|y)$$

In other words, the guided model determines the update direction using a simple linear combination of the conditional and unconditional score functions. Classifier-free guidance then tends to show better results than classifier guidance, because the built Bayesian classifier is much more robust than a separately trained one. Moreover, it does not require an auxiliary model, and diffusion models can be made compatible with classifier-free guidance simply through conditioning dropout during training [Die23]. An illustration of classifier-free guidance is shown in figure 1.

3 Proposed method for generating audio loops with conditioning using IADB

3.1 UNet

We recall that our goal is to generate conditioned audio loops with IADB. To do so, we rely on the UNet architecture [RB15] for our de-blending network. A UNet network basically follows and Encode-Decode architecture :

The Encoder generates a condensed representation of the input by diminishing its dimension while retaining pertinent information. Comprising a series of convolution layers, each block reduces the input size while doubling the number of channels.

The Decoder reconstructs and interprets the deep features, transforming them into the final representation through similar blocks of deconvolution layers.

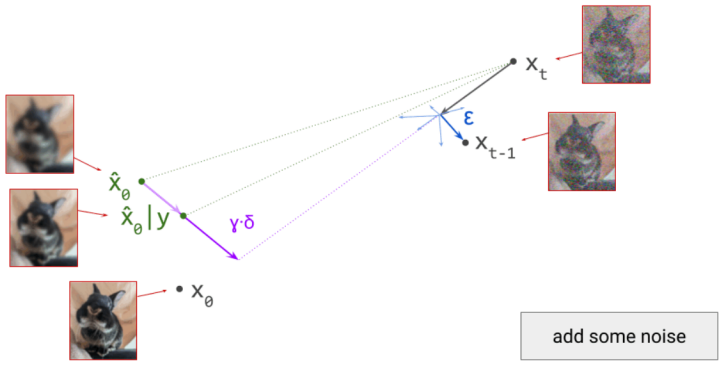

Figure 1: Illustration of a diffusion denoising step (t to $t-1$) using classifier-free guidance [Die23].

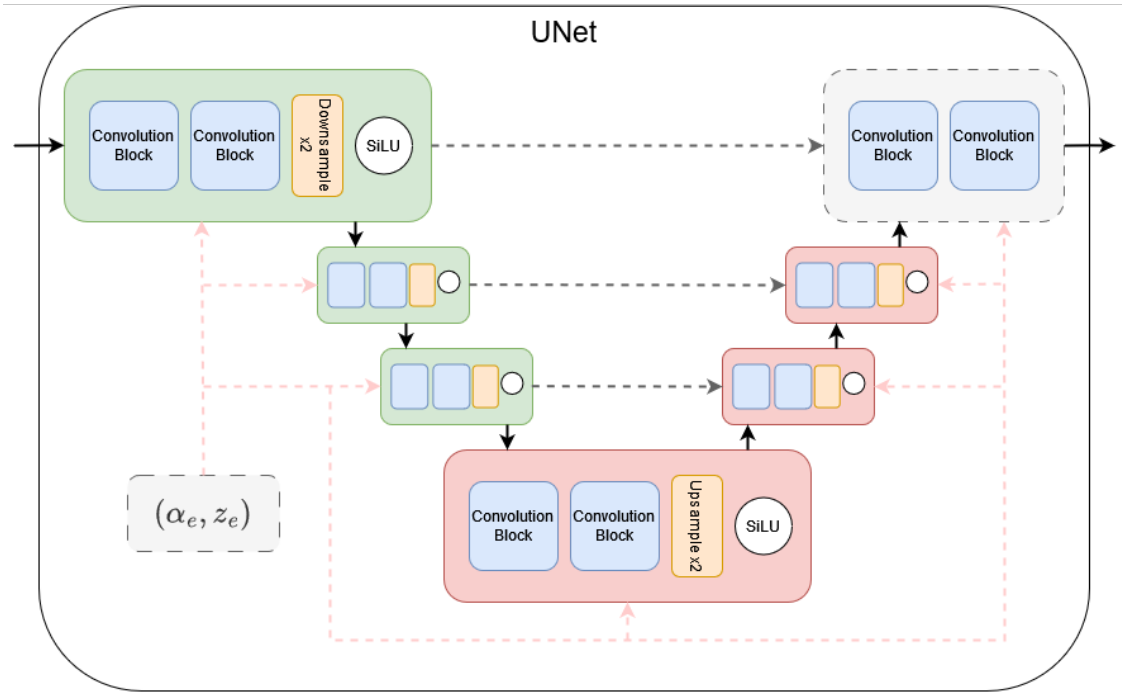

Figure 2: General model's architecture with UNet

To enhance the symmetry between the encoder and decoder, the U-Net incorporates skip-connections. These connections link layers of the encoder and decoder at corresponding depths, refining the reconstruction process by progressively conveying more detailed information from the encoder to the decoder. Specifically, feature maps of a layer in the encoder are concatenated with their equivalent counterparts in the decoder.

The relevancy of such an architecture for our work comes from both the use of convolutions which are particularly suited for audio applications and the skip connections, that allow the network to keep track of high-level information in each layer.

Our implementation is similar to the one presented in the original article [RB15], with 4 encoder blocks, each one composed of 2 sub-blocks of 1D-convolutions with 3x3 filters, batch normalization and and a Sigmoid Linear Unit (SiLU) activation. The decoder has a mirror-like architecture and uses similar sub-blocks for deconvolutions.

2

3.2 Conditioning with CLAP and audio descriptors

3.2.1 Conditioning using semantic-aware normalization AdaGN

A new normalization method, called Adaptive Group Normalization (AdaGN) and introduced by [ZFZ+20], allows for the learning of semantic features and their relative importance along the feature channels. This is done by first defining groups along the channels based on the semantic similarities of the channel dimensions. Then, by conducting a group normalization and by projecting latent vectors into a space that recognizes the semantics along the feature channels, we can control the importance of the different semantics by controlling their statistics via the projected latent codes. In our implementation, we separate the semantic embedding into two different embeddings α_e and z_e, the first representing the iteration within the denoising process and the second representing any stationary latent embedding (stationary for the audio frames and not the denoising steps). In our model, α_e is obtained with a Sinusoidal Positional Encoding (SPE) [VSP+17] followed by a feed-forward layer, and z_e is obtained by passing text or audio through the corresponding CLAP encoder [WCZ+23] followed by a feed-forward layer. Given g groups, the formulation of AdaGN is given by

$$\mathbf{Y} = \sum_{0 \leq i < g} AdaGN(\alpha_{\mathbf{e}}^{\mathbf{i}}, \mathbf{z}_{\mathbf{e}}^{\mathbf{i}}, \mathbf{X}_i) = \sum_{0 \leq i < g} z_e^i (\alpha_e^{i,a} \frac{\mathbf{X}_i - \mu(\mathbf{X}_i)}{\sigma(\mathbf{X}_i)} + \alpha_e^{i,b})$$

where $\alpha_{\mathbf{e}}^{\mathbf{i}}$ and $\mathbf{z}_{\mathbf{e}}^{\mathbf{i}}$ are the i^{th} element of the embeddings α_e and z_e and $\alpha_e^i = (\alpha_e^{i,a}, \alpha_e^{i,b})$. This formulation is equivalent to the original AdaGN formulation by combining α_e and z_e.

3.2.2 Conditioning with audio descriptors

As music is an information evolving through time, conditioning can also be done with continuous audio-descriptors that represent the sample behaviour such as its beat, centroid or loudness. First step is to extract data from a given descriptor, and then convert it into a new representation that can be integrated in our learning process, typically a signal evolving over the 1024 frames we decided to convert time in. A relevant way to give our network the conditioning information is to concatenate our descriptor representation with the latent codes of our sample.

Hence, in order to extract our conditioning information at different scales of encoding, we create an embedding of our conditioning signal using a 1D-convolution with a stride of size 2. We repetitively do this operation to reduce its dimension to the same size as latent code we are concatenating it with on the second convolution block of each encoding/decoding step. Finally, we use a dropout function on this additional information to make our model use the classifier-free guidance results and improve our training [DN21].

3.3 EnCodec

To train our model efficiently, we rely on EnCodec, an autoencoder model trained specifically to allow for high-fidelity audio compression [DCSA20]. This is particularly useful for us, as we can train our model with the latent representation of our preferred audio dataset through EnCodec's encoder and apply IADB consequently. Since our model also outputs latent code, we have to decode it in an audio format using EnCodec's decoder. 3

4 Experiments

4.1 2D Examples

To appreciate the simplicity of IADB over a diffusion probabilistic model, we would first use this architecture on a 2D dataset. By training a simple Multi-Layer Perception (MLP) to denoise a 2D distribution, new distributions could then be generated with the sampling procedure, recreating the results of the authors of IADB in [HBC23]. 4

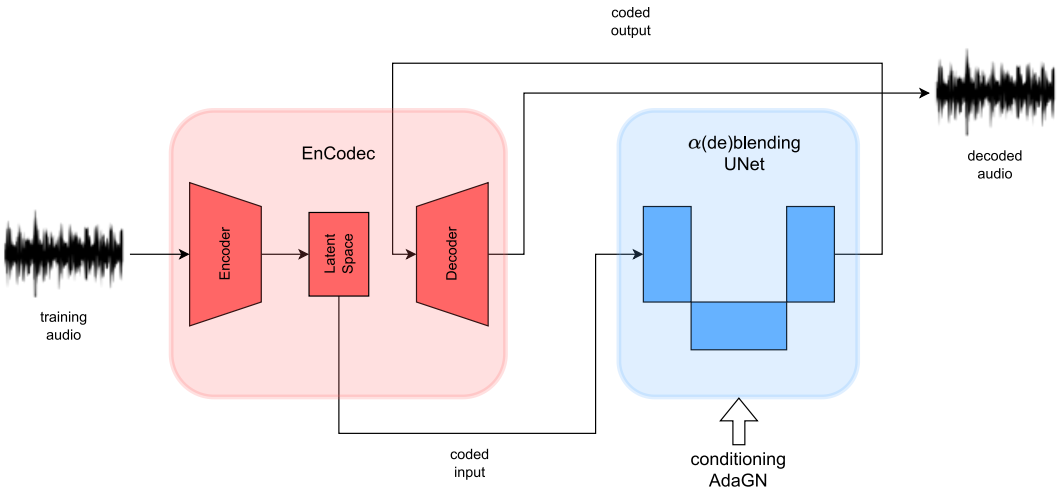

Figure 3: Training setup with IADB and EnCodec for conditioned audio loop generation

Figure 4: Deblending process on a 2D point cloud to generate the ACIDs IRCAM logo

4.1.1 Audio Dataset for audio loop generation with IADB

To generate coherent music loops with time efficiency and expressive controls, the choice of the dataset on which we should train our model and the quality of the decoding process is crucial. To begin with, we think it is a good idea to restrain the music genres present in the training data to a limited amount. In doing so, we ensure that our model learns the specific characteristics of the curated music style and more deeply fits the boundaries that define it. It also allows us to give our model a simpler conditioning approach by choosing controls that can have quantitative effects on loop generation.

In this particular experiment, we extract features and samples from a personally protected dataset composed of minimal electronic house and techno music. The main reason (beyond our appreciation of this music genre) is that this music has historically been built on repetitive rhythmic patterns and percussive elements, which are faster to infer with diffusion models than melodies on which classical or pop music is based. To do so, we train the EnCodec [DCSA20] model specifically on our curated dataset to get the latent code corresponding to its representation, but we also go through a feature extractor process giving us interesting audio descriptors that can be used for conditioning such as the harmonic key or the beat-tracking signature.

Finally, we use a trained CLAP [WCZ+23] checkpoint to extract latent text and audio representations from our audio loops.

4.1.2 Results for BPM Conditioning

As already mentioned, the particularity of the dataset we have for the training is the presence of repetitive rhythmic sequences, of which tempo is by definition a core element. Hence, using a beat-tracking audio descriptor to condition our diffusion model seems particularly appropriate to our creative goals.

To get a learnable representation of rhythm, we use a beat-tracking information extracted from our audio samples metadata during pre-processing. This data is stored in an array that gives us a sequence of times in seconds characterizing each sample. In order to make the information continuous

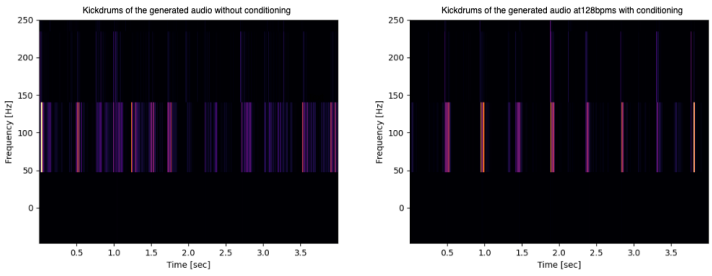

Figure 5: Generated audio spectrograms (left: without conditioning. right: with a 128bpm conditioning). Frequencies are cut to 250Hz to only keep the information relative to the kickdrum of the generated audio, which eventually defines the rhythm of the sequence.

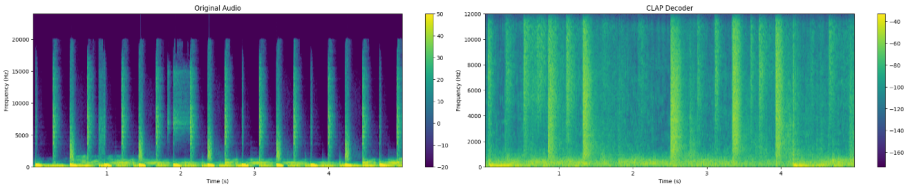

Figure 6: Spectrograms of diffusion with CLAP conditioning (left: original audio, right: audio conditioned with CLAP).

and relevant for conditioning, we apply a simple transform and convert this array into a consistent signal that can be concatenated with its corresponding sample at the different U-Net steps. A good representation we can transform the array into is a saw-tooth-like signal, which rises linearly from 0 to 1 between each time step. Hence, it injects a strong discontinuity for each new beat and still keeps track of its temporal position between two of them.

Inference is done using a similar process. To get a more musically coherent output, we condition the input of our model with a constant BPM value that we transform into time signatures then a saw-tooth signal. Results are unequivocal as we can easily hear and verify on spectrograms the influence of our conditioning on our audio outputs.5.

4.2 Results for conditioning with CLAP

When training the model on conditioning with CLAP latent codes, the model learns to map the CLAP dual-modal latent space to the EnCodec latent space. Because EnCodec has both an encoder and a decoder, the combined process of deblending and decoding can be viewed as a decoder for the CLAP latent space. This can be used as a text-to-music process or can be very useful for applying transformations directly in the CLAP latent space. The results for decoding CLAP latent codes produce a considerable amount of noise but capture the main percussive and rhythmic structure of the audio 6. These results are encouraging for future work.

5 Applications

5.1 Time-stretching

A powerful property of the α-(de)Blending method used for diffusion is that it is a deterministic process [HBC23]. Indeed, we can hope to inverse this process and determine the vector sampled from the distribution p_0 given a latent vector sampled from our dataset distribution p_1.This inverse diffusion process can allow us to find x_0 given x_1 and then re-apply the original diffusion process with conditioning. By using a BPM condition on this last diffusion process, we hope to obtain the same original sample, but in a different BPM. In other words, we hope to obtain a time-stretched version of the original sample x_1. Another way of achieving a time-stretched sample would be to apply the diffusion process on a random sample $x_0 \sim p_0$ and conditioning the process with our wanted BPM and with the CLAP latent vector of the sample x_1 we wish to time-stretch.

5.2 Track transition

Given two tracks, we may want to find a smooth and musically coherent transition between them. Because we learned a pseudo-decoder for the CLAP latent space, we can infer a sample while interpolating between the CLAP vectors of two different tracks. This should create a transition between them. However, this is a prior impossible because conditioning except the BPM is stationary for an audio time. A possible solution to this problem would be to split the audio generation into n smaller generated frames, each with a stationary conditioning embedding within that frame. The final audio is constructed with the sequence of frames with overlapping and window scaling. Therefore, the conditioning can vary from frame to frame. By combining this method with the previous time-stretching method, we can transition between both BPM of the tracks.

5.3 Loop generation

These experiments show us that it is possible to generate music loops with expressive control over a specified BPM range. Hence it provides a powerful tool for music production and allows the user to express creation with a different approach from the way it is usually done. The generated loops can be used for sampling, pattern matching, or in any other way we could think of. Moreover, we think about enhancing the beat tracking control by adding some sub-beat information to it, so we could use this type of control in a similar way as a sequencer. More details on descriptor-based controls can be found in this article [DDN$^+$23].

6 Conclusion

In this paper, we propose a concrete and novel application of the simple and deterministic diffusion approach Iterative α-(de)Blending [HBC23] to generate audio samples. Our conditioning experiments, which rely on the classifier-free-guidance approach [HS22], show that it is possible and efficient to use relevant audio descriptors such as BPM to control our generative process, and open the path for some future work possibilities on text-to-music work with CLAP [WCZ$^+$23]. Indeed, by training a deblending model conditioned on CLAP, we can in a way decode CLAP latent codes into waveform. However, our performance using CLAP is quite poor. This can be remedied with a more diversified dataset, more training time, and a more complex or larger model. Furthermore, we observed greater consistency in our inference results for BPM falling between 120 and 130. However, outside this range, our model encounters challenges in applying learned features, leading to a lack of qualitative musical sample generation. To address this issue effectively, a straightforward solution would be to enhance the dataset diversity by incorporating a broader range of musical sub-genres, which would eventually contain slower and faster samples than the ones we have. Finally, as this paper shows the efficiency of conditioning our model with audio features via the example of BPM, it would be interesting to find more descriptors to condition our generative process with, in order to provide different types of control.

References

[DCSA20] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *preprint arXiv:2210.13438*, 2020.

[DDN$^+$23] Ninon Devis, Nils Demerlé, Sarah Nabi, David Genova, and Philippe Esling. Continuous descriptor-based control for deep audio synthesis, 2023.

[Die23] Sander Dieleman. The geometry of diffusion guidance, 2023.

[DN21] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021.

[HBC23] Eric Heitz, Laurent Belcour, and Thomas Chambon. Iterative alpha-(de)blending: a minimalist deterministic diffusion model. *arXiv preprint arXiv:2305.03486*, 2023.

[HS22] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[RB15] Olaf Ronneberger and Philipp Fischerand Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241, 2015.

[RM15] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. *ICML Conference*, 2015.

[SDWMG15] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv:1503.03585v8 [cs.LG]*, 2015.

[SYL15] Kihyuk Sohn, Xinchen Yan, and Honglak Lee. Learning structured output representation using deep conditional generative models. *NeurIPS Proceedings*, 2015.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[WCZ+23] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[ZFZ+20] Heliang Zheng, Jianlong Fu, Yanhong Zeng, Jiebo Luo, and Zheng-Jun Zha. Learning semantic-aware normalization for generative adversarial networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21853–21864. Curran Associates, Inc., 2020.