Adding temporal musical controls on top of pretrained generative models

Sarah Nabi, Nils Demerlé, Geoffroy Peeters, Frédéric Bevilacqua, Philippe Esling

ADDING TEMPORAL MUSICAL CONTROLS ON TOP OF PRETRAINED GENERATIVE MODELS

Sarah Nabi[*1] **Nils Demerlé**[*1] **Geoffroy Peeters**[2] **Frédéric Bevilacqua**[1] **Philippe Esling**[1]
[1] UMR 9912 STMS-IRCAM, Sorbonne Université, CNRS, Paris, France
[2] LTCI, Télécom-Paris, Institut Polytechnique de Paris, France
sarah.nabi@ircam.fr, nils.demerle@ircam.fr

ABSTRACT

Recent advances in deep generative modeling have enabled high-quality models for musical audio synthesis. However, these approaches remain difficult to control, confined to simple, static attributes and, most importantly, entail retraining a different computationally-heavy architecture for each new control. This is inefficient and impractical as it requires substantial computational resources.

In this paper, we propose a novel approach allowing to add time-varying musical controls on top of any pretrained generative models with an exposed latent space (e.g. neural audio codecs), without retraining or finetuning. Our method supports both discrete and continuous attributes by adapting a rectified flow approach with a latent diffusion transformer. We learn an invertible mapping between pretrained latent variables and a new space disentangling explicit control attributes and *style* variables that capture the remaining factors of variation. This enables both feature extraction from an input, but also editing those features to generate transformed audio samples. Finally, this also introduces the ability to perform synthesis directly from the audio descriptors. We validate our method with 4 datasets going from different musical instruments up to full music recordings, on which we outperform state-of-the-art task-specific baselines in terms of both generation quality and accuracy of the control by inferring transferred attributes. Our code is available on the supporting webpage [1].

1. INTRODUCTION

Since the pioneering autoregressive model WaveNet [1], significant advances have been made in deep generative modeling for raw audio waveform synthesis. In particular, neural audio codecs [2–4] recently enabled fast high-quality audio generation with sampling rates up to 48kHz. These models directly compress the raw audio waveform into a temporal *latent space* that captures high-level audio features. However, these latent representations are difficult to interpret and highly entangled, precluding their use as straightforward controls [5]. To address this, existing methods usually rely on explicit conditioning techniques, bypassing the latent information [6, 7], adding adversarial regularization [8], or learning an additional model to condition the latent codes [9–11]. However, these approaches are often limited in the types of controls usable [6, 9] and, most importantly, they require to retrain [8, 11] or finetune the underlying large synthesis model [10], which is inefficient and impossible for many as it requires substantial amount of data and computational resources.

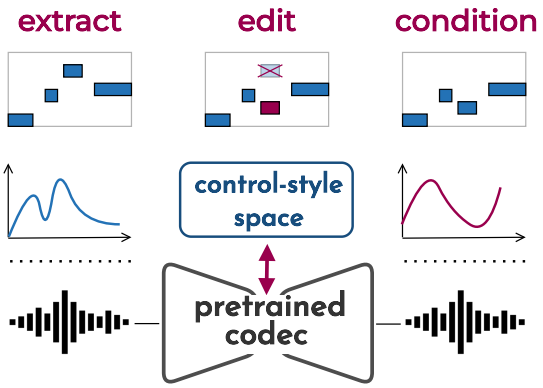


Figure 1: Our approach enhances pretrained neural codecs with a new *control-style* space supporting both discrete and continuous time-varying musical controls, enabling features extraction, audio editing and conditional synthesis.

Although neural codecs address the technical difficulties of modeling high-dimensional audio waveforms, the remaining challenge is to find interpretable control over the latent variables, which are usually restrained to follow a simple Gaussian prior distribution [2]. In this paper, we aim to give the ability to the end user to define its own controls by reshaping the structure of the pretrained latent space. Hence, we propose a new method to add temporal musical controls on top of any pretrained generative model with an exposed latent space without requiring retraining nor finetuning. As depicted in Fig. 1, our method supports both discrete and continuous features, allowing to combine any types of control including both static and

[*]Equal contribution
[1]https://acids-ircam.github.io/platune/

time-varying ones such as instrument labels, MIDI inputs or continuous descriptors. Furthermore, our method is able to be trained very efficiently on top of any frozen pretrained codec by adapting the PluGeN [5] approach. This method proposed to transform the entangled latent representation of pretrained generative models into a multi-dimensional *control-style* space where latent features are splited between *control* variables modeling the selected attributes and *style* variables capturing the remaining factors of variations. It uses discrete labels to parameterize the control distributions and relies on Normalizing Flows [12] (NFs) to learn an invertible mapping between both spaces. Although PluGeN provides the ability to add new controls on pretrained synthesis models, it is limited to static label attributes and was only applied for conditional generation of static images. Here, we enhance pretrained neural audio codecs with a new *control-style* space by extending the PluGeN approach to time-varying musical controls. As dealing with audio signals implies more complex temporal latent spaces and NFs suffer from training instability and lack scalability with strong architecture constraints, we also propose a more efficient way to define the mapping between both spaces by adapting the deterministic diffusion-based rectified flow approach [13] to temporal latent spaces using a Latent Diffusion Transformer [14].

Thanks to the invertible mapping, our proposed method becomes highly versatile and enables (1) *feature extraction*, (2) *input audio editing*, as well as (3) *explicit conditional synthesis*, as illustrated in Fig 1. We benchmark our model on each of these 3 tasks with 4 increasingly complex datasets composed of recordings ranging from monophonic musical instruments up to full music pieces. We show that our model efficiently extracts and improves the control of multiple complex audio features compared to state-of-the-art baselines, while maintaining the generation quality and enabling conditioning with unseen combinations of attributes.

2. BACKGROUND

2.1 Diffusion models

Deep generative models aim to model the underlying distribution $p(\mathbf{x})$ of a given dataset. Diffusion Models (DMs) achieve this by corrupting the data with noise and then learning to reverse this process through denoising. Given a series of noise levels $\sigma_0 < \sigma_1 < ... < \sigma_N$, we can define a set of mollified distributions $p_{\sigma_t}(\mathbf{x}_t) = \int p(\mathbf{x})\mathcal{N}(\mathbf{x}_t; \mathbf{x}, \sigma_t^2 I)d\mathbf{x}$, obtained by adding i.i.d Gaussian noise to the data. Although many formulations of diffusion exists, a common approach is to train a denoiser network D_θ, typically a neural network to minimize the following denoising score matching objective

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\mathbb{E}_{\mathbf{n}\sim\mathcal{N}(0,\sigma^2 I)}\|D(\mathbf{x}+\mathbf{n};\sigma)-\mathbf{x}\|_2^2. \quad (1)$$

The model implicitly learns to approximate the score of each perturbed distribution $s_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_{\sigma_t}(\mathbf{x}) = \frac{D(\mathbf{x};\sigma)-\mathbf{x}}{\sigma^2}$ and hence can generate new samples using annealed Langevin dynamics. Starting from $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, it follows the score functions towards the data by iteratively updating $\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{T} * \nabla_{\mathbf{x}} \log p_{\sigma_t}(\mathbf{x}_t)$ until converging to a sample $\mathbf{x}_0$ from $p(\mathbf{x})$. DMs currently achieve state-of-the-art results in image [15–17] and audio generation [18, 19], offering high-quality synthesis with stable training and strong conditioning abilities.

2.2 Rectified flow

Rectified Flow (RF) [13] is a recent proposal that builds upon diffusion models and optimal transport ideas, aiming to directly learn transport maps between data distributions. Unlike diffusion models, which are constrained to a Gaussian noise prior, RFs generalize to any distribution and have demonstrated superior performance in latent generation tasks [20]. Considering two arbitrary distributions π_0 and π_1, RF constructs a deterministic continuous-time flow by learning an ordinary differential equation (ODE) model that connects observations ($\mathbf{x}_0 \sim \pi_0, \mathbf{x}_1 \sim \pi_1$) through straight paths. Formally, the rectified flow induced from ($\mathbf{x}_0$, $\mathbf{x}_1$) is defined by the ODE

$$d\mathbf{z}_t = v(\mathbf{z}_t, t)dt, \quad (2)$$

where $\mathbf{z}_t$ represents the data point at time $t \in [0, 1]$[2], and $v : \mathbb{R}^d \to \mathbb{R}^d$ is the *drift force*, typically parameterized by a neural network which is trained to minimize

$$\min_v \int_0^1 \mathbb{E}\left[\|v(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2\right] dt, \quad \text{with} \quad \mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1. \quad (3)$$

This objectives encourages the flow to follow straight paths between $\mathbf{x}_0$ and $\mathbf{x}_1$, leading to an efficient and invertible deterministic mapping between π_0 and π_1.

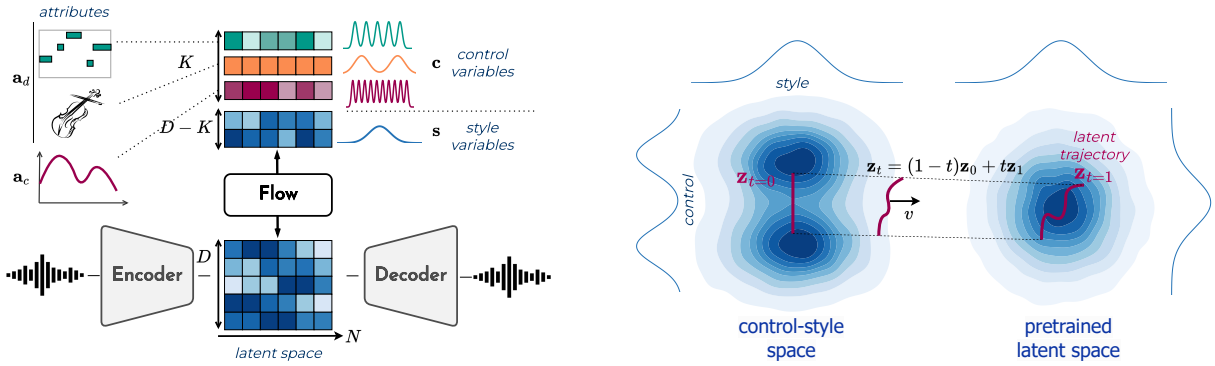
2.3 Neural audio codecs

Neural audio codecs such as RAVE [2], EnCodec [3], or Music2Latent [4] enable fast high-quality waveform synthesis. By leveraging autoencoder schemes, it directly compresses the input audio signal into a time series of latent variables $\mathbf{z} \in \mathbb{R}^{D\times N}$ (where D and N are respectively the embedding and time dimensions) defined in a lower-dimensional *latent space*. In addition to their computational efficiency, these models are also powerful representation learning frameworks. During training, the *encoder* captures high-level features from the input data, while the *decoder* learns to reconstruct the original waveform from its latent representation. However, these latent variables are very difficult to interpret and highly entangled which precludes straightforward control over these models.

2.4 Control of deep generative models

To provide explicit controls, prior works rely on conditioning techniques [21, 22] and directly use the audio features as additional training inputs to the synthesis model [6, 7].

[2] refers to the parametrization in the ODE and not musical time



(a) Adding musical controls on pretrained neural audio codecs.

(b) Mapping controls to latent trajectories using rectified flow.

Figure 2: Detailed overview of our approach mapping pretrained latent codes to explicit control and style variables.

FaderRave [8] proposed to condition RAVE [2] with non-differentiable time-varying descriptors using an adversarial criterion [22], but requires to retrain the codec from scratch. Recently, latent diffusion prior models were also introduced to achieve efficient conditional generation [16, 17]. The AFTER [11] model provides explicit controls and example-based style transfer on neural codecs by adapting the DiffAE [23] approach. They train two semantic encoders with a latent diffusion model to disentangle global timbre properties and time-varying musical structure. However, most existing approaches are limited in the types of control they provide [6, 9] and require to retrain [7, 8] or fine-tune [10] the synthesis model.

Recently, PluGeN [5] enhanced pretrained generative models with multi-label attributes conditioning to control image generation without retraining. It transforms the entangled latent representation into a multi-dimensional space disentangling attributes across individual dimensions. This new space is composed of *control* variables $\mathbf{c}$, defined as independent one-dimensional Gaussian mixture distributions, and *style* variables $\mathbf{s}$ supposed to capture the remaining factors of variations. To learn an invertible mapping between the two spaces, PluGeN relies on Normalizing Flows [12] (NFs) preserving the dimensionality. Each example $\mathbf{x} \in \mathbb{R}^{d_x}$ is paired with discrete multi-labels $\mathbf{y} = (y_1, ..., y_K) \in [\![1, M_k]\!]^K$, where M_k is the number of classes for each attribute y_k. The flow transforms the latent code $\mathbf{z} \in \mathbb{R}^D$ into $(\mathbf{c}, \mathbf{s}) = (c_1, ..., c_K, s_1, ..., s_{D-K}) \in \mathbb{R}^D$ such that the target distribution corresponds to

$$p_{\mathbf{C},\mathbf{S}|\mathbf{Y}=\mathbf{y}}(\mathbf{c}, \mathbf{s}) = \prod_{k=1}^{K} p_{C_k|Y_k=y_k}(c_k) \cdot p_{\mathbf{S}}(\mathbf{s}), \quad (4)$$

where *style* variables are modeled by a standard Gaussian

$$p_{\mathbf{S}} = \mathcal{N}(\mathbf{0}, \mathbf{I}_{D-K}), \quad (5)$$

and *control* variables with a mixture of Gaussians

$$p_{C_k|Y_k=y_k} = \prod_{i=1}^{M_k} \mathcal{N}(\mu_i, \sigma_i^2)^{1_{y_k=i}}, \quad (6)$$

where μ_i are evenly distributed between -1 and 1, σ_i are user-defined parameters, and $1_{y_k=i}$ is the indicator function which equals 1 if $y_k = i$ and 0 otherwise.

3. PROPOSED METHOD

Our approach aims to add temporal musical controls on top of any pretrained generative model with an exposed latent space, without requiring retraining. We adapt PluGeN [5] for time-varying attributes to create a new space that disentangles explicit control features from the remaining *style* variations. We introduce a rectified flow model to learn an invertible mapping between the *control-style* space and the pretrained latent space, which directly *transports* the control signals into its latent trajectory. Once trained, we can use this space to extract musical features, edit audio samples by shifting the control trajectories or transferring attributes, and perform explicit conditional synthesis. Our overall approach is depicted in Fig. 2.

3.1 Defining the time-varying control-style space

The training data $\mathbf{x} \in \mathbb{R}^{d_x}$ are paired with K attributes $\mathbf{a} = (\mathbf{a}_d, \mathbf{a}_c) \in \mathbb{R}^{K \times N}$, where $\mathbf{a}_d$ and $\mathbf{a}_c$ refer to *discrete* and *continuous* features, resampled to align with the latent trajectory $\mathbf{z} = E(\mathbf{x}) \in \mathbb{R}^{D \times N}$. As in eq. 6, we parameterize the control distributions as time-evolving mixtures of Gaussians for each control variable $c_k(n)$ which leads to

$$p_{c_k|a_{dk}(n)}(c_k, n) = \prod_{i=1}^{M_k} \mathcal{N}(c_k|\mu_k^{(i)}(n), \sigma_k^2)^{1_{a_{dk}(n)=i}}, (7)$$

for discrete attributes, where ${\mu_k}^{(i)}$ corresponds to the mean of the i-th class control distribution, uniformly distributed between -1 and 1 for all M_k classes.

For continuous ones, we consider

$$p_{c_k|a_k(n)}(c_k, n) = \mathcal{N}(c_k|\mu_k(n), \sigma_k^2), \quad (8)$$

where the means $\mu_k(n)$ correspond to the attribute values normalized between -1 and 1 [3]. We sample the remaining style variables $\mathbf{s} \in \mathbb{R}^{(D-K) \times N}$ from a standard Gaussian as in eq. 5 and define the *control-style* space with the joint distribution

$$p_{\mathbf{c},\mathbf{s}|\mathbf{a}}((\mathbf{c}, \mathbf{s}), n) = \prod_{k=1}^{K} p_{c_k|a_k(n)}(c_k, n) \cdot p_{\mathbf{s}}(\mathbf{s}, n). \quad (9)$$

[3] minimum and maximum values of each attributes $\mathbf{a}_{min}$ and $\mathbf{a}_{max}$ are computed for the whole dataset.

3.2 Mapping temporal latent and control distributions

As depicted in Fig. 2b, we rely on a rectified flow to directly map the temporal control-style space to the pretrained latent space. We use eq. 9 to define the input distribution π_0 from Sec. 2.2, and sample a control-style trajectory $\mathbf{z}_{t=0} = (\mathbf{c}, \mathbf{s}) \sim p_{\mathbf{c},\mathbf{s}|\mathbf{a}}$. We use the associated latent trajectory $\mathbf{z} = \mathbf{z}_{t=1} \sim p_{\mathbf{z}}$ as the target signal. Then, we uniformly sample a time step $t \sim \mathcal{U}(0,1)$ to compute the intermediate temporal representation

$$\mathbf{z}_t = (1-t) \times \mathbf{z}_{t=0} + t \times \mathbf{z}_{t=1}. \tag{10}$$

We parametrize the drift force of the flow v_θ with a Latent Diffusion Transformer (DiT) [14] and optimize

$$\mathcal{L}_\theta = \mathbb{E}_{\substack{t\sim\mathcal{U}(0,1) \\ \mathbf{z}_{t=0}\sim p_{\mathbf{c},\mathbf{s}|\mathbf{a}} \\ \mathbf{z}_{t=1}\sim p_{\mathbf{z}}}} \left\| v_\theta(\mathbf{z}_t, t) - (\mathbf{z}_{t=1} - \mathbf{z}_{t=0}) \right\|^2. \tag{11}$$

The full training procedure is described in Algorithm 1.

Algorithm 1 Training procedure for our method

Input: data $\mathbf{x} \in \mathcal{D}$, attributes $\mathbf{a} = (\mathbf{a}_d, \mathbf{a}_c)$, control variances $\sigma_{\mathbf{c}}^2$, encoder E of the pretrained audio codec, denoising diffusion model v_θ, learning rate γ
for each training step **do**
  Encode $E(\mathbf{x}) = \mathbf{z} = [\mathbf{z}_1, ..., \mathbf{z}_N] = \mathbf{z}_{t=1}$
  Align $\mathbf{a} = [\mathbf{a}_1, ..., \mathbf{a}_N]$
  Compute $\mu_{\mathbf{c}} = 2 \times \frac{\mathbf{a}-\mathbf{a}_{\min}}{\mathbf{a}_{\max}-\mathbf{a}_{\min}} - 1$
  Define $p_{\mathbf{c}} = \mathcal{N}(\mu_{\mathbf{c}}, {\sigma_{\mathbf{c}}}^2)$
  Sample control $\mathbf{c} \sim p_{\mathbf{c}}$
  Sample style $\mathbf{s} \sim p_{\mathbf{s}} = \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_{t=0} = (\mathbf{c}, \mathbf{s})$
  Sample time step $t \sim \mathcal{U}(0,1)$
  Compute $\mathbf{z}_t = (1-t) \times \mathbf{z}_{t=0} + t \times \mathbf{z}_{t=1}$
  Compute $\mathcal{L}_\theta = \mathbb{E}\left[||v_\theta(\mathbf{z}_t, t) - (\mathbf{z}_{t=1} - \mathbf{z}_{t=0})||^2\right]$
  Update $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}_\theta$
end for
return ϵ_θ

3.3 Inference

Once the deterministic mapping from controls to latents is learned, we can leverage the disentangling properties of our method to infer the attributes $\hat{\mathbf{a}}$ of a given audio sample $\mathbf{x}$ in each pre-defined control dimension by reversing the diffusion process to predict $\hat{\mathbf{z}}_{t=0} = (\hat{\mathbf{c}}, \mathbf{s})$ from $\mathbf{z}_{t=1} = E(\mathbf{x})$ by applying

$$\mathbf{z}_{t-1} = \mathbf{z}_t - v_\theta(\mathbf{z}_t, t) \times dt. \tag{12}$$

for T sampling steps, where $dt = \frac{1}{T}$. We can then edit the sample's features by directly manipulating the control variables or perform conditional synthesis by sampling and mapping the new $(\tilde{\mathbf{c}}, \mathbf{s})$ representation to the pretrained latent space through

$$\mathbf{z}_{t+1} = \mathbf{z}_t + v_\theta(\mathbf{z}_t, t) \times dt. \tag{13}$$

Finally, we use the pretrained decoder D to reconstruct the waveform $\tilde{\mathbf{x}} = D(\tilde{\mathbf{z}}_{t=1})$.

4. EXPERIMENTS

We aim to assess the control abilities of our proposed method for our 3 target tasks, namely, *musical features retrieval*, *audio editing*, and *conditional synthesis*. We evaluate our approach on time-varying and static attributes, both discrete and continuous, on 4 datasets ranging from monophonic instruments up to full music recordings. We provide audio examples on the supporting webpage [4].

4.1 Datasets

Monophonic instruments We benchmark our 3 tasks on the **URMP** [24] and **MedleySolos** [25] datasets composed of musical pieces played by diverse instruments. We retained only the monophonic instruments with 13 categories for URMP and 6 for MedleySolos, resulting in 2,600 and 13,000 examples of 6 and 3 seconds respectively. We define the *instrument* label as a control dimension and use the MIDI information extracted with BasicPitch [26] to control the *melody* defined by pitch, octave, onsets and dynamics features each encoded in individual dimension. We also use continuous descriptors, namely, *centroid*, *bandwidth*, *integrated loudness* [5] and higher-level timbral features, *sharpness* and *booming* [6], computed frame-wise directly from audio samples. Hence, we evaluate on both discrete and continuous, static and time-varying attributes.

Polyphonic instrument We rely on the **MaestroV3** [25] dataset composed of piano recordings of classical pieces. We split the audio files into 12-second chunks, resulting in 27,000 examples. Using the *music21* library [7], we compute high-level MIDI descriptors that characterise the playing techniques and melodic content, namely *average note duration*, *note density*, *central pitch* and *pitch range*. To obtain time-varying features, we compute the descriptors on a 4 second sliding window with a 0.5 second hop size.

Full music Finally, we evaluate our method on the large-scale open source **Jamendo** [27] dataset, containing a diverse collection of musical recordings, covering various genres and labeled with emotion tags. To evaluate our model on high-level perceptual descriptors, we rely on Music2Emotion [28] [8], a pretrained emotion recognition model trained on the same dataset. Using a simple cross-correlation analysis we retain four relatively independent tags among the most represented in the dataset : "dark", "fast", "emotional", "children". We use the classification output of Music2Emotion for those 4 tags, computed on a sliding window of 10 seconds to obtain a time-varying measure of each emotion. We normalize and resample those features to obtain our control signals.

4.2 Evaluation metrics

To evaluate our method, we assess both its audio synthesis quality, but also its ability to correctly extract features, and

[4] https://acids-ircam.github.io/platune/
[5] computed with https://github.com/csteinmetz1/pyloudnorm
[6] https://github.com/AudioCommons/timbral_models
[7] https://www.music21.org/
[8] https://github.com/AMAAI-Lab/Music2Emotion

		Extract		Quality (FAD) ↓			Onset F1 score ↑			Instrument Acc. (%) ↑		
		mel↓	inst↑	Rec.	Edit	Synth.	Rec.	Edit	Synth	Rec.	Edit	Synth.
Medley S.	AFTER [11]	-	-	0.308	0.330	0.415	0.422	0.355	0.322	**91.7**	76.9	-
	PluGeN [5]	**0.127**	**99.26**	1.267	0.847	1.271	0.002	0.026	0.002	12.54	36.85	12.23
	Ours	0.208	94.32	**0.188**	**0.203**	**0.216**	**0.427**	**0.375**	**0.386**	91.28	**79.97**	**82.72**
URMP	AFTER [11]	-	-	**0.235**	**0.271**	0.402	**0.536**	**0.487**	**0.441**	**82.0**	**57.8**	-
	PluGeN [5]	**0.078**	**86.46**	1.516	0.758	1.518	0.001	0.121	0.007	8.59	21.88	8.59
	Ours	0.192	58.53	0.265	0.318	**0.304**	0.521	0.294	0.433	77.34	28.13	36.71

Table 1: Comparison of different models for the *features extraction*, *editing* and *conditioning* tasks with melody and instrument controls. MSE on the normalized attributes is used for melody extraction and accuracy for instrument. Lower FAD indicates better quality, while higher onset F1 score and Instrument Accuracy indicate better performance.

resynthesize audio with accurately transformed features.

Features retrieval We compare the attributes inferred by the model with the groundtruth attributes by using the Mean Square Error (MSE) on the normalized control dimensions, except for the instrument where we compute a prediction accuracy score of belonging to the correct Gaussian of the mixture.

Attributes editing and transfer We extract the control-style representations from the audio inputs and randomly swap the attribute that we aim to evaluate. For instance, we keep the melody and associated style representation but update the instrument dimension with a random pick among the available categories. This enables us to assess the disentanglement properties of our method by also giving unseen combinations of attributes. First, we evaluate the impact of this change over the generation quality using the Frechet audio Distance [29] (FAD) on the CLAP [30] embeddings computed with the generated waveforms. Then, we evaluate the accuracy of control by extracting the updated attribute from the generated waveform and compare it to the target attribute. For the *melody* control, we extract the MIDI with BasicPitch [26] and compare it using the Onset F1 score from *mir-eval*, where two notes are considered identical if the pitch and onsets are the same within $\pm$50ms of each other. For instrument prediction, we trained a classifier on CLAP embeddings of the training set, and use it to predict an accuracy score. For continuous attributes, we simply use the MSE as mentioned above.

Conditional synthesis We evaluate conditional synthesis by randomly selecting a combination of ground truths controls and sampling a style representation from the standard Gaussian distribution to generate a new waveform. We evaluate the synthesis quality with FAD and compute the control precision similarly to editing.

4.3 Implementation details

We rely on Music2Latent [4] as pretrained neural codec, using the the official implementation and weights [9] that compresses waveforms sampled at 44.1kHz into 64-dimensional time-varying latent codes with a time compression ratio of 4096. The model architecture is a Latent DiT [14] with 15M parameters for the monophonic instrument experiments, scaled up to 27M parameters for Maestro and Jamendo datasets. All models are trained for 300k steps using AdamW [32] optimizer with a constant learning rate of 1e-4.

4.4 Baselines

PluGeN [5] The approach most closely related to our work is PluGeN which can be evaluated on the 3 tasks similarly. However, it was applied only to static attributes for image data. To use it as a straightforward fair baseline that takes into account the time-varying features, we adapt the WaveGlow [33] architecture on top of the latent space of the pretrained Music2Latent removing the conditioning on mel-spectrograms. This enables to directly map the temporal latent trajectory to the control-style space with affine coupling flow layers as in the original paper.

AFTER [11] We train the *midi-to-audio* configuration of AFTER using the most recent available implementation [10]. For instrument editing we use the timbre embedding of a randomly sampled example matching the target label, and for synthesis we randomly sample the timbre space.

SDEdit [31] To compare our approach with an existing editing strategy, we train a conditional rectified flow model $v_\theta(\mathbf{z}_t, c, t)$ to map samples between a Gaussian noise distribution $\tilde{\pi}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the latent data distribution π_1. To perform editing, we first invert a data sample $(\mathbf{z}_1, c)$ to its corresponding noise $\mathbf{z}_0$ (which can be seen as analoguous to the style vector $\mathbf{s}$ from our proposal) following $\mathbf{z}_{t-1} = \mathbf{z}_t - v_\theta(\mathbf{z}_t, c, t) \times dt$. Then, we generate an edited version with controls c_{swap} through the denoising process $\mathbf{z}_{t+1} = \mathbf{z}_t + v_\theta(\mathbf{z}_t, c_{swap}, t) \times dt$ starting from $\mathbf{z}_0$.

5. RESULTS

5.1 Monophonic instruments

First, we evaluate our method on the 3 tasks for monophonic instruments (MedleySolos and URMP datasets) and present our results in Table 1 and Table 2.

[9] https://github.com/SonyCSLParis/music2latent

[10] https://github.com/acids-ircam/AFTER

	Quality (FAD) ↓			Onset F1 score ↑			Inst. Acc. (%) ↑			Cont. Feat. MSE ↓		
	Rec.	Edit	Synth.	Rec.	Edit	Synth.	Rec.	Edit	Synth.	Rec.	Edit	Synth.
SDEdit [31]	**0.131**	**0.154**	**0.289**	**0.552**	**0.264**	**0.267**	**93.8**	36.2	38.5	**0.042**	**0.048**	**0.058**
PluGeN [5]	1.248	0.946	1.239	0.001	0.007	0.001	15.60	25.38	15.90	0.310	0.329	0.305
Ours	0.201	0.211	0.319	0.374	0.144	0.110	**88.07**	**51.68**	**58.56**	0.061	0.070	0.079

Table 2: Comparison on MedleySolos for editing and conditioning tasks combining both discrete and continuous controls.

5.1.1 Features retrieval

Although PluGeN generally achieves better results for the features extraction task, our method remains competitive on MedleySolos and manages to extract the correct instrument from the original waveform. We believe that the lower results on URMP are due to the lack of data in the training set, making it more difficult for the model to generalize to the 3 tasks simultaneously. Moreover, the Normalizing Flows [12] training objective of PluGeN explicitly penalize the model on the control accuracy while the feature extraction is implicitly learned with our method.

5.1.2 Audio Editing

As shown in Table 1, our method outperforms both PluGeN and AFTER on Medley Solos for independent melody and instrument editing. This demonstrates the strong disentanglement abilities of our approach, that remains robust to unseen combinations of attributes created by swapping randomly the target control dimension. This is especially the case for the instrument control with high prediction accuracy, which implies that all the instrument information is well-encoded in the control variable and not the style representation. On the contrary, PluGeN completely fails to follow the target controls, indicating that most of the information is encoded in the *style* dimension. Furthermore, our method ensures good generation quality and achieves the lowest FAD score on MedleySolos on all scenarios. However, AFTER demonstrates better performance on URMP, especially in melody control, which might also suggest a minimum amount of data required by our method.We present our results for the combination of melody, instrument and audio descriptors in Table 2. Although SDEdit achieves better FAD and melody control, our method achieves a better instrument control, indicating that some features are difficult to edit with simple conditioning. Introducing descriptors control degrades the melody accuracy, which can be explained by correlations between controls like pitch contour and centroid.

5.1.3 Conditional synthesis

We now evaluate our model on conditional synthesis, for both paired (Rec.) and unpaired (Synth.) attributes. Our method still outperfoms AFTER on Medley Solos. Interestingly, the gap between paired and unpaired melody accuracy seems to be greater for AFTER, suggesting that our method provides a better disentanglement between attributes, independently of the global conditioning strength. This is also supported by the stronger increase in FAD for AFTER, while our model maintains more consistent audio quality across configurations.

		Extract	Generation		
		MSE ↓	FAD ↓	Edit ↓	Synth. ↓
Maestro	SDEdit [31]	-	**0.008**	0.364	0.362
	Ours	0.267	0.021	**0.340**	**0.325**
Jamedo	SDEdit [31]	-	**0.054**	0.068	0.060
	Ours	0.079	0.105	**0.047**	**0.033**

Table 3: Comparison on the Jamendo and MaestroV3 datasets. Control accuracy is averaged across features.

5.2 Application to high-level controls

We present our results on MaestroV3 and Jamendo datasets for high-level control tasks in Table 3. Our approach consistently outperforms SDEdit, with particularly significant improvements on Jamendo, which focuses on the challenging task of emotion-based editing. The underlying principle of conditional diffusion models is that the conditioning signal provides additional information during training to guide the denoiser and reduce the denoising loss. However, these models tend to exhibit lower performance when dealing with high-level features that have complex, non-trivial relationships with the data. In contrast, our method explicitly learns an invertible mapping between the control and data spaces, improving the control efficiency over high-level attributes, on top of providing control extraction capabilities. Finally, qualitative analysis reveals that, for both datasets, the style vector effectively captures the majority of the melodic content and, for Jamendo, even the lyrics and orchestration. This property enables users to generate meaningful variations of a given track or even create hybrids between emotions or playstyles.

6. CONCLUSION

We presented a new efficient method to add controls on pretrained neural audio codecs. To the best of our knowledge, this is the first approach defining an invertible mapping between arbitrary controls and audio latent codes enabling a large variety of applications. We leave for future works, improvements on the low shot performance, aiming to provide more personalized controls on small datasets for creative applications.

7. ACKNOWLEDGMENTS

This work has been supported by the Paris Ile-de-France Région in the framework of DIM AI4IDF.

8. REFERENCES

[1] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, vol. 12, 2016.

[2] A. Caillon and P. Esling, "Rave: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[3] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.

[4] M. Pasini, S. Lattner, and G. Fazekas, "Music2latent: Consistency autoencoders for latent audio compression," in *International Society for Music Information Retrieval, ISMIR 2024*, 2024.

[5] M. Wołczyk, M. Proszewska, Ł. Maziarka, M. Zieba, P. Wielopolski, R. Kurczab, and M. Smieja, "Plugen: Multi-label conditional generation from pre-trained models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8647–8656.

[6] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "Ddsp: Differentiable digital signal processing," *arXiv preprint arXiv:2001.04643*, 2020.

[7] Y. Wu, E. Manilow, Y. Deng, R. J. Swavely, K. Kastner, T. Cooijmans, A. Courville, A. Huang, and J. Engel, "Midi-ddsp: Hierarchical modeling of music for detailed control," in *Proceedings of the Tenth International Conference on Learning Representations (ICLR)(Online)(2022 Apr.)*, 2022.

[8] N. Devis, N. Demerlé, S. Nabi, D. Genova, and P. Esling, "Continuous descriptor-based control for deep audio synthesis," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[9] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 47 704–47 720, 2023.

[10] S.-L. Wu, C. Donahue, S. Watanabe, and N. J. Bryan, "Music controlnet: Multiple time-varying controls for music generation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 2692–2703, 2024.

[11] N. Demerlé, P. Esling, G. Doras, and D. Genova, "Combining audio control and style transfer using latent diffusion," in *International Society for Music Information Retrieval, ISMIR 2024*, 2024.

[12] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*. PMLR, 2015, pp. 1530–1538.

[13] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[14] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4195–4205.

[15] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," *Advances in neural information processing systems*, vol. 35, pp. 26 565–26 577, 2022.

[16] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[17] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[18] Z. Evans, C. Carr, J. Taylor, S. H. Hawley, and J. Pons, "Fast timing-conditioned latent audio diffusion," in *Forty-first International Conference on Machine Learning*, 2024.

[19] Q. Huang, D. S. Park, T. Wang, T. I. Denk, A. Ly, N. Chen, Z. Zhang, Z. Zhang, J. Yu, C. Frank *et al.*, "Noise2music: Text-conditioned music generation with diffusion models," *arXiv preprint arXiv:2302.03917*, 2023.

[20] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *Forty-first international conference on machine learning*, 2024.

[21] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.

[22] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, "Fader networks: Manipulating images by sliding attributes," *Advances in neural information processing systems*, vol. 30, 2017.

[23] K. Preechakul, N. Chatthee, S. Wizadwongsa, and S. Suwajanakorn, "Diffusion autoencoders: Toward a meaningful and decodable representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 619–10 629.

[24] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2018.

[25] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," in *International Conference on Learning Representations*.

[26] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, "A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Singapore, 2022.

[27] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The mtg-jamendo dataset for automatic music tagging," in *Machine learning for music discovery workshop, international conference on machine learning (ICML 2019)*, 2019, pp. 1–3.

[28] J. Kang and D. Herremans, "Towards unified music emotion recognition across dimensional and categorical models," 2025. [Online]. Available: https://arxiv.org/abs/2502.03979

[29] D. Roblek, K. Kilgour, M. Sharifi, and M. Zuluaga, "Fr\'echet audio distance: A reference-free metric for evaluating music enhancement algorithms," in *Proc. Interspeech*, 2019, pp. 2350–2354.

[30] 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," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[31] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," in *International Conference on Learning Representations*.

[32] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[33] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3617–3621.