Disentangling Multi-instrument Music Audio for Source-level Pitch and Timbre Manipulation

Yin-Jyun Luo[1]* **Kin Wai Cheuk**[2] **Woosung Choi**[2] **Wei-Hsiang Liao**[2]
Keisuke Toyama[3] **Toshimitsu Uesaka**[2] **Koichi Saito**[2] **Chieh-Hsin Lai**[2] **Yuhta Takida**[2]
Simon Dixon[1] **Yuki Mitsufuji**[2,3]
[1]C4DM, Queen Mary University of London [2]Sony AI [3]Sony Group Corporation
`yin-jyun.luo@qmul.ac.uk` `kinwai.cheuk@sony.com`

Abstract

Disentangling pitch and timbre from the audio of a musical instrument involves encoding these two attributes as separate latent representations, allowing the synthesis of instrument sounds with novel attribute combinations by manipulating one representation independently of the other. Existing solutions have mostly focused on single-instrument audio, excluding the cases where multiple sources of instruments are presented. To fill the gap, we aim to disentangle multi-instrument mixtures by extracting per-instrument representation that combines the pitch and timbre latent variables. These latent variables construct a set of modular building blocks that is used to condition a decoder to compose new mixtures. We first present a simple implementation to verify the framework using structured and isolated chords. We then scale up to a complex dataset of four-part chorales by a model that jointly learns the latents and a diffusion transformer. Our evaluation identifies the key components for the success of disentanglement and demonstrates the application of mixture transformation based on source-level attribute manipulation. Audio samples are available at `https://yjlolo.github.io/dismix-audio-samples`.

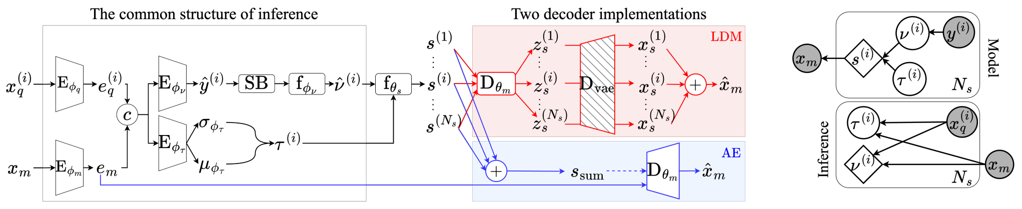
1 Introduction

Disentangled representation learning is concerned with learning semantically meaningful features of observed data in a compact latent space [1]. Most of the existing work applies generative frameworks such as variational autoencoders (VAEs) [26], which encodes data by associating factors of variation with independent subspaces of the latent space and reconstructs the data given the encoding [50, 6, 22, 18]. Manipulating a subspace while leaving the rest unchanged leads to sparse variation of particular factors (to which the manipulated subspace corresponds) in the decoder output [17], and thus facilitates applications such as controllable transformation of the original data.

The technique has been applied to music audio to disentangle representations of timbre (e.g., the musical instrument played in a recording) and pitch (e.g., the melody played by the instrument) [33, 9, 49, 34, 48, 35, 11, 3, 54, 30, 55]. It allows for transferring the instrument (or melody) played in a reference to a target instrument (or melody) provided by another example by swapping the timbre (or pitch) representation, while preserving the melody (or instrument) originally played in the reference.

Despite its wide adoption for instrument attribute transfer, prior arts are limited in that the analysis and synthesis of these attributes are only amenable to a single instrument at a time, thereby excluding scenarios where multiple instruments are present in the audio. Although MSI-DIS [28] addresses two-instrument mixtures, the model is designed to generate single-instrument outputs instead of mixtures, preventing it from capturing the interdependence of constituent instruments.

*This work was conducted during an internship at Sony AI.


The common structure of inference
Two decoder implementations
LDM
AE
SB
Model
Inference
N_s

(a) The framework (b) The graphical illustration

Figure 1: (a) **The framework**, DisMix, extracts for each ith source a source-level latent $s^{(i)}$ combining a timbre $\tau^{(i)}$ and a pitch latent $\nu^{(i)}$, given a query $x_q^{(i)}$ and a mixture x_m composed of N_s sources of instruments. As the framework is not limited to accepting a fixed set of instruments, the timbre of the query dictates the target instrument to extract from the mixture. A decoder, either a simple autoencoder (AE) or a latent diffusion model (LDM), reconstructs the mixture given a set of N_s source-level latents. (b) **The graphical models of the generative process (top) and the inference network (bottom)**. Diamond nodes denote deterministic mappings of their parent nodes.

To bridge the gap, we propose *DisMix*, a framework that disentangles a mixture of instruments and composes new mixtures based on novel combinations of pitch and timbre representations. DisMix represents each instrument by a source-level representation that combines a pitch and a timbre latent variable. To encode these variables, the encoders are conditioned on a mixture and a set of queries. The timbre of a query determines which instrument's latents are extracted from the mixture. The pitch and timbre latents are encoded in separate subspaces, allowing them to be independently manipulated per instrument. A decoder then takes as input the manipulated set of source-level representations and renders a new mixture consisting of sources whose pitch and timbre are dictated by the manipulation.

The pitch and timbre representations act as modular building blocks to construct the melody and instrument of a source or an "audio object" [4]. The decoder assembles these objects in a way that preserves their pitch and timbre. This is reminiscent of "object representation" [16] motivated by the human ability to understand complex ideas in terms of reusable and primitive components [14].

Contributions In summary: (**i**) We propose DisMix (Fig. 1a) as a framework to decompose a multi-instrument mixture into a set of per-instrument pitch and timbre representations, and we showcase its applications in mixture composition and editing by manipulating the representations in a modular manner; (**ii**) We first demonstrate a simple implementation of DisMix as a proof of concept (Section 4.1), and, by implementing a latent diffusion model (LDM) [42], scale up to a complex dataset featuring four-part Bach Chorales played by a pool of 13 orchestral instruments (Section 4.2); (**iii**) We propose to use a binarisation layer and verify it to be crucial for pitch-timbre disentanglement. We also show that reconstructing a mixture by capturing interdependencies among the constituent sources outperforms iterative single-source reconstruction in both disentanglement and audio quality.

2 Related Work

Pitch and Timbre Disentanglement While existing work focuses on single-instrument input [33, 11, 3, 54, 12, 49, 48, 9, 34, 9, 35, 30, 55, 36], some studies extract pitch and timbre information from multi-instrument mixtures for tasks other than representation learning [20, 8, 7]. Closer to our work, MSI-DIS [28] addresses the transcription, separation, and generation of individual sources from mixtures. We build upon MSI-DIS but target the task of mixture composition and editing.

Object-Centric Representation Learning Object representations encode physical objects in a visual scene using unique representations [16, 31, 46, 56, 57]. For audio, MusicSlots [15] and AudioSlots [44] learn entangled representations of constituent sources from mixtures, while we disentangle pitch and timbre of sources and tackle a more complex dataset with a conditional LDM.

3 DisMix: The Proposed Framework

Given x_m, a mixture composed of N_s unique sources of musical instruments $\{x_s^{(i)}\}_{i=1}^{N_s}$, our goal is to extract for each instrument a source-level latent $s^{(i)}$ that combines representations of timbre $\tau^{(i)}$ and pitch $\nu^{(i)}$. These N_s source-level latents are used to condition a decoder to reconstruct the mixture.

We consider DisMix in Fig. 1a as a backbone framework that takes as input both a mixture and a set of queries to extract the corresponding set of source-level latents. We present two implementations: A simple autoencoder (AE) that serves as a proof of concept (Section 4.1) and an LDM (Section 4.2) that scales to a complex dataset. They share a common structure for encoding the pitch and timbre latents and differ in how they recover a mixture from a set of source-level latents.

Generative Process Fig. 1b (top) shows that the generative process samples x_m conditioned on a set of source-level representations $\mathcal{S} = \{s^{(i)}\}_{i=1}^{N_s}$, where $s^{(i)} = \mathrm{f}_{\theta_s}(\tau^{(i)}, \nu^{(i)})$ is a deterministic function of timbre and pitch. $\tau^{(i)}$ is sampled from a prior $\mathcal{N}\big(0, \mathrm{I}\big)$, while $\nu^{(i)}$ is conditionally sampled from a distribution parameterised by its pitch annotation $y^{(i)}$. The joint distribution can be written as:

$$p_{\theta_m}(x_m, \{\tau^{(i)}\}_{i=1}^{N_s}, \{\nu^{(i)}\}_{i=1}^{N_s} | y^{(i)}) = p_{\theta_m}(x_m|\mathcal{S}) \prod\nolimits_{i=1}^{N_s} p_{\theta_\nu}(\nu^{(i)}|y^{(i)}) p(\tau^{(i)}), \tag{1}$$

where $\nu^{(i)}$ and $\tau^{(i)}$ are sampled independently, whereby the pitch and timbre latents can act as modular building blocks to compose a mixture, facilitating source-level pitch-timbre disentanglement and manipulation demonstrated in Section 4.

We specify the networks and their parameters (i.e., θ_m, θ_s, and θ_ν) for AE and LDM in Sections 4.1 and 4.2, respectively. Note that Fig. 1a omits the conditional prior $p_{\theta_\nu}(\nu^{(i)}|y^{(i)})$ and we investigate the effect of the prior for the AE while excluding it from the LDM for simplicity.

Learning DisMix is by introducing an inference network (Section 3.1) to approximate posteriors over the latent variables that are otherwise intractable, and by maximising a lower bound to the marginal likelihood $p(x_m|\{y^{(i)}\}_{i=1}^{N_s})$ [26]. The training objective is described in Section 3.2.

3.1 Inference Network

Our inference network in Fig. 1b (bottom) extracts the timbre and the pitch latent variable given both a mixture and a query, motivated by the query-based source separation framework [28, 27, 52]. A query $x_q^{(i)}$ shares the timbre characteristics with a constituent source $x_s^{(i)}$ (unseen during evaluation) of the mixture – both are played by the same instrument, regardless of pitch. We assume that the instrument identities of the constituent sources are known and the queries are available.

Each mixture x_m is paired with a set of N_s queries randomly sampled from the dataset. We extract the features $e_m = \mathrm{E}_{\phi_m}(x_m)$ and $e_q^{(i)} = \mathrm{E}_{\phi_q}(x_q^{(i)})$, where $\mathrm{E}_{\phi_m}(\cdot)$ and $\mathrm{E}_{\phi_q}(\cdot)$ are neural networks. We let $e_{m,q}^{(i)} \coloneqq \{e_m, e_q^{(i)}\}$ to avoid clutter for the rest of the section.

Pitch Encoder As shown in Fig. 1b, we propose that the pitch and timbre encoders take a common factorised form: $q_{\phi_u}(\mathcal{U}|x_m, \{x_q^{(i)}\}_{i=1}^{N_s}) = \prod_{i=1}^{N_s} q_{\phi_u}(u^{(i)}|e_{m,q}^{(i)})$, where $u \in \{\nu, \tau\}$ and $\mathcal{U} \in \{\{\nu^{(i)}\}_{i=1}^{N_s}, \{\tau^{(i)}\}_{i=1}^{N_s}\}$. Given $e_{m,q}^{(i)}$, the pitch and timbre latents of the ith source are encoded independently of each other and of other sources. To promote disentanglement, we apply a binarisation layer [10, 13] to constrain the pitch latent and prevent it from capturing excessive information.

As shown in Fig. 1a, we extract the pitch latent by steps of transcription, binarisation, and translation:

$$\text{Transc. } \hat{y}^{(i)} = \mathrm{E}_{\phi_\nu}(e_{m,q}^{(i)}); \text{ Binarise } \hat{y}_{\text{bin}}^{(i)} = 1_{\{\text{Sigmoid}(\hat{y}^{(i)}) > h\}}; \text{ Translate } \hat{\nu}^{(i)} = f_{\phi_\nu}(\hat{y}_{\text{bin}}^{(i)}). \tag{2}$$

Both $\mathrm{E}_{\phi_\nu}(\cdot)$ and $\mathrm{f}_{\phi_\nu}(\cdot)$ are neural networks. $\mathbf{1}_{\{.\}}$ is the indicator function that defines the binarisation layer SB and $h \sim \mathrm{U}(0,1)$ is a threshold sampled for each training step and is fixed at 0.5 during the evaluation. The straight-through estimator [2] is used to bypass the non-differentiable operator. Altogether, we define the pitch encoder as follows:

$$q_{\phi_\nu}(\nu^{(i)}|e_{m,q}^{(i)}) \coloneqq \delta(\nu^{(i)} - \mathrm{f}_{\phi_\nu}(\hat{y}_{\text{bin}}^{(i)})), \tag{3}$$

where $\delta(\cdot)$ is the Dirac delta function and $\nu^{(i)}$ deterministically takes the value of $\hat{\nu}^{(i)}$ in Eq. (2).

Timbre Encoder The timbre encoder parameterises the mean and variance of a Gaussian posterior:

$$q_{\phi_\tau}(\tau^{(i)}) = \mathcal{N}(\tau^{(i)}; \mu_{\phi_\tau}(e_{m,q}^{(i)}), \sigma_{\phi_\tau}^2(e_{m,q}^{(i)})\mathrm{I}), \tag{4}$$

where $q_{\phi_\tau}(\tau^{(i)}) \coloneqq q_{\phi_\tau}(\tau^{(i)}|e_{m,q}^{(i)})$. The timbre latent is sampled from the Gaussian by reparameterisation trick [26], i.e., $\tau^{(i)} = \mu_{\phi_\tau}(\cdot) + \epsilon\sigma_{\phi_\tau}(\cdot)$, where $\epsilon \sim \mathcal{N}\big(0, \mathrm{I}\big)$. We let $p(\tau^{(i)}) = \mathcal{N}\big(0, \mathrm{I}\big)$ to constrain the information capacity of the timbre latent through the Kullback–Leibler divergence (KLD) in $\mathcal{L}_{\text{ELBO}}$ (5).

3.2 Training Objectives

Evidence Lower Bound We maximise $\mathcal{L}_{\text{ELBO}}$, an evidence lower bound (ELBO) to the marginal log-likelihood $\log p(x_m|\{y^{(i)}\}_{i=1}^{N_s})$ (we provide the derivation in Appendix A):

$$\underbrace{\mathbb{E}_{\prod_i q_{\phi_\tau}(\tau^{(i)})}\big[\log p_{\theta_m}(x_m|\mathcal{S})\big]}_{\text{mixture reconstruction}} + \sum\nolimits_i \underbrace{\log p_{\theta_\nu}(\hat{\nu}^{(i)}|y^{(i)})}_{\text{pitch prior}} - \underbrace{\mathcal{D}_{\text{KL}}\big(q_{\phi_\tau}(\tau^{(i)})\|p(\tau^{(i)})\big)}_{\text{timbre prior}}, \quad (5)$$

where $\hat{\nu}^{(i)} = \text{f}_{\phi_\nu}(\hat{y}_{\text{bin}}^{(i)})$, $\mathcal{S} = \{s^{(i)}\}_{i=1}^{N_s}$, and $s^{(i)} = \text{f}_{\theta_s}(\tau^{(i)}, \hat{\nu}^{(i)})$ as described previously. We specify θ_m, θ_s, and θ_ν in Sections 4.1 and 4.2. We also have access to individual sources during training, so we add to $\mathcal{L}_{\text{ELBO}}$ the sum of source-wise likelihood terms $\sum_{i=1}^{N_s} \mathbb{E}_{q_{\phi_\tau}(\tau^{(i)})}\big[\log p_{\theta_m}(x_s^{(i)}|s^{(i)})\big]$. Note that we reuse the decoder with the parameters θ_m for source reconstruction. Again, we exclude the pitch prior, associated with the second term in $\mathcal{L}_{\text{ELBO}}$ (5), from the LDM (Section 4.2) for simplicity, and report its effect on disentanglement for the AE in Appendix C.1.

Pitch Supervision To enhance the disentanglement, we also minimise a binary cross entropy loss $\text{BCE}(\hat{y}^{(i)}, y^{(i)})$ where $y^{(i)}$ is the pitch annotation of the ith source. We explore relaxing the model by excluding this term from the LDM and report the results in Table 2.

Barlow Twins We minimise a simplified Barlow Twins loss [58] to enhance the correlation between the query and the timbre latent for them sharing the timbre characteristics: $\mathcal{L}_{\text{BT}} = \sum_{i=1}^{N_s}\sum_{d=1}^{D_\tau}(1 - \mathcal{C}_{dd}(e_q^{(i)}, \tau^{(i)}))^2$, where $\mathcal{C}$ is a cross-correlation matrix, and both $e_q^{(i)}$ and $\tau^{(i)}$ share the same dimensionality D_τ. Empirically, $\mathcal{L}_{\text{BT}}$ counteracts the over-regularisation effect of the timbre prior and promotes a discriminative timbre space, as shown in Fig. 5.

In summary, we maximise $\mathcal{L}_{\text{DisMix}} = \mathcal{L}_{\text{ELBO}} - \mathcal{L}_{\text{BCE}} - \mathcal{L}_{\text{BT}}$ and do not find it necessary to explicitly weigh each loss term. Next, we describe the implementations and results of the AE and the LDM.

4 Experiments and Results

4.1 A Simple Case Study Using an Autoencoder

$p_{\theta_m}(x_m|\mathcal{S})$ in $\mathcal{L}_{\text{ELBO}}$ (5) is a Gaussian likelihood parameterised by a decoder $\text{D}_{\theta_m}(\cdot)$, which can be a permutation-invariant function that outputs the Gaussian mean, reconstructing a mixture given a set of source-level representations. We opt for a simpler decoder here and discuss a transformer in Section 4.2. In particular, we slightly deviate from $\mathcal{L}_{\text{ELBO}}$ (5) and reconstruct x_m using e_m and s_{sum} during training and evaluation, respectively, as illustrated by the solid and dashed blue arrows in Fig. 1a. This introduces an additional loss term which we detail and discuss in Appendix B.1.

MusicSlots The dataset [15] synthesises 3,131 unique chords from JSB Chorales [5] using sound fonts of piano, violin, and flute. Each composite note of a chord is synthesised by a sound font randomly sampled with replacement, whereby a sound font can play multiple notes ($N_s \in \{1,2,3\}$). These notes together define a source, annotated by a multi-hot vector $y^{(i)} \in \{0,1\}^{N_p}$ and $N_p = 52$. The note waveforms are summed to form the chord waveform which defines a mixture. There are 28,179 samples of mixtures divided into training, validation, and test sets with a ratio of 70/20/10.

Evaluation We encode pitch and timbre latents of each mixture from the test set and randomly permute the latents across sources to obtain novel source-level latents. The sum of the source-level latents is then decoded, indicated by the dashed blue arrow in Fig. 1a. We use pre-trained pitch and instrument classifiers to verify if the decoder manifests the pitch and timbre after the permutation. We detail the evaluation protocol and study the effect of pitch priors of different complexities (the second term in $\mathcal{L}_{\text{ELBO}}$ (5)) in Appendix C.1. An ablation study of other loss terms is reported in Table 1.

Results Table 1 suggests that both the standard Gaussian prior and the SB layer are key components for disentanglement. Minimising $\mathcal{L}_{\text{BT}}$ prevents the standard Gaussian prior from over-regularising the timbre latent, and dropping the loss term negatively affects pitch accuracy for mixture rendering.

4.2 A Latent Diffusion Framework

We implement an LDM framework [45], where $p_{\theta_m}(x_m|\mathcal{S})$ in $\mathcal{L}_{\text{ELBO}}$ (5) is parameterised by a diffusion transformer (DiT) [42]. As illustrated by the red arrows in Fig. 1a, the DiT reconstructs

Table 1: AE: An ablation study for various loss terms measured by classification accuracy (%).

	Disentanglement		Mix. Rendering	
	Pitch	Inst.	Pitch	Inst.
DisMix	93.39	100.00	90.69	100.00
- $\mathcal{L}_{\text{BT}}$	93.18	99.92	87.92	100.00
- KLD	69.41	100.00	35.10	100.00
- SB	93.46	46.71	40.23	98.91

Table 2: LDM: Disentanglement and audio quality in terms of classification accuracy (%) and Fréchet Audio Distance (FAD), respectively.

	Instrument (↑)		Pitch (↑)		FAD (↓)	
	Set	Single	Set	Single	Set	Single
M0	89.94	**96.43**	94.37	**97.15**	2.27	**2.13**
M1	**97.49**	96.05	97.15	97.02	**2.10**	2.14
M2	82.83	80.16	**97.19**	97.03	2.14	2.14

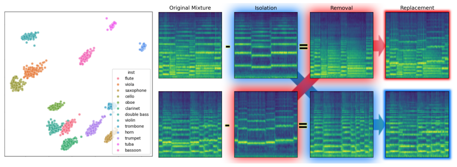


Figure 2: *Left*: PCA of the timbre space. *Right*: Source swaps between two mixtures.

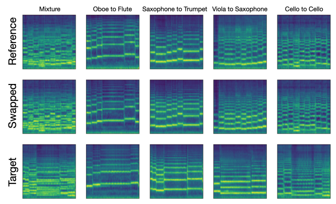


Figure 3: Instrument replacement of a reference mixture given a target mixture.

features $\{z_s^{(i)}\}_{i=1}^{N_s}$ given $\{s^{(i)}\}_{i=1}^{N_s}$. $z_s^{(i)}$ is a compact feature of the source $x_s^{(i)}$ extracted by the pre-trained VAE encoder in AudioLDM2 [29]. The sources can then be recovered by decoding the compact features using $\mathrm{D}_{\text{vae}}(\cdot)$, the AudioLDM2 decoder. Summing the recovered sources in the audio domain yields the mixture. We leave the implementation details of the DiT in Appendix B.2.

CocoChorale The dataset [53] provides realistic four-part chorales ($N_s = 4$) in the style of Bach Chorales [5], featuring 13 orchestral instruments. We use the random ensemble subset, which amounts to 60,000 chorales, totalling 350 hours of audio. We sample four-second segments from the chorales, and for each part the pitch annotation $y^{(i)} \in \{0,1\}^{N_p \times T_p}$, where $N_p = 129$ and $T_p = 400$. The dataset follows a split ratio of 80/10/10 for the training, validation, and test sets.

Evaluation Similar to Section 4.1, the evaluation involves pre-trained classifiers (see Appendix C.2). In Table 2, M0 is trained to reconstruct a single source conditioned on a source-level latent; it iterates N_s times to recover a mixture. This is a proxy for MSI-DIS [28] despite the difference between our strategies for disentanglement and model architectures. M1 is the proposed set-conditioned model, and M2 drops the pitch supervision. All models can output N_s sources (thus a mixture) in both single-source ("Single") and set-conditioned ("Set") manners, regardless of how they were trained, as the DiT accepts variable length input. We also report the average FAD [21] over 13 instruments by comparing the VGG features of the reconstructed sources with that of the original data.

Results M1 yields the best overall performance with "Set", the mode the model was trained, which suggests the benefit of modelling the interdependencies over the sources [38, 39]. The superior instrument accuracy is consistent with the descriminative timbre latent space shown on the left side of Fig. 2. Surprisingly, M0 performs decently under "Set", although it was not trained using the mode, which could imply that our DiT adaptation is effective. M2 performs reasonably well even without pitch supervision, indicating that SB imposes an effective bottleneck. Removing pitch supervision leaks excessive information to the pitch latent and hurts the instrument classification accuracy.

Figs. 2 and 3 are examples of pitch and timbre latents acting as modular building blocks for the composition of the mixture. The right side of Fig. 2 exemplifies how we can "exchange" sources between two mixtures by swapping their source-level latents (i.e., both pitch and timbre), while Fig. 3 replaces constituent instruments (i.e., only timbre) of a reference with those in a target mixture.

5 Conclusion

We present DisMix to disentangle multi-instrument mixtures into modular building blocks of pitch and timbre representations for compositional manipulations of mixtures. The requirement of queries is a limitation that we plan to address in future work. We are also interested in "attribute inpainting" where the conditional decoder is asked to complete missing pitch and timbre in its conditions.

References

[1] Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. In *Trans. on Pattern Analysis and Machine Intelligence*, 2013.

[2] Y. Bengio, N. Léonard, and A. Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*, 2013.

[3] A. Bitton, P. Esling, and A. Chemla-Romeu-Santos. Modulated Variational Auto-encoders for Many-to-many Musical Timbre Transfer. *arXiv preprint arXiv:1810.00222*, 2018.

[4] J. K. Bizley and Y. E. Cohen. The What, Where and How of Auditory-object Perception. *Nature Reviews Neuroscience*, 2013.

[5] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In *Int. Conf. on Machine Learning*, 2012.

[6] R. T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating Sources of Disentanglement in Variational Autoencoders. In *Conf. on Neural Information Processing Systems*, 2018.

[7] K. W. Cheuk, K. Choi, Q. Kong, B. Li, M. Won, J.-C. Wang, Y.-N. Hung, and D. Herremans. Jointist: Simultaneous Improvement of Multi-Instrument Transcription and Music Source Separation via Joint Training. *arXiv preprint arXiv:2302.00286*, 2023.

[8] F. Cwitkowitz, K. W. Cheuk, W. Choi, M. A. Martínez-Ramírez, K. Toyama, W.-H. Liao, and Y. Mitsufuji. Timbre-trap: A Low-resource Framework for Instrument-agnostic Music Transcription. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2024.

[9] O. Cífka, A. Ozerov, U. Şimşekli, and G. Richard. Self-Supervised VQ-VAE for One-Shot Music Style Transfer. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2021.

[10] H.-W. Dong and Y.-H. Yang. Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation. In *Int. Soc. for Music Information Retrieval*, 2018.

[11] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In *Int. Conf. on Machine Learning*, 2017.

[12] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton. Generative Timbre Spaces with Variational Audio Synthesis. In *Int. Conf. on Digital Audio Effects*, 2018.

[13] J. Fajtl, V. Argyriou, D. Monekosso, and P. Remagnino. Latent Bernoulli Autoencoder. In *Int. Conf. on Machine Learning*, 2020.

[14] J. A. Fodor and Z. W. Pylyshyn. Connectionism and Cognitive Architecture: A Critical Analysis. *Cognition*, 1988.

[15] J. Gha, V. Herrmann, B. Grewe, J. Schmidhuber, and A. Gopalakrishnan. Unsupervised Musical Object Discovery from Audio. In *Conf. on Neural Information Processing Systems, ML4Audio Workshop*, 2023.

[16] K. Greff, S. Van Steenkiste, and J. Schmidhuber. On the Binding Problem in Artificial Neural Networks. *arXiv preprint arXiv:2012.05208*, 2020.

[17] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner. Towards a Definition of Disentangled Representations. *arXiv preprint arXiv:1812.02230*, 2018.

[18] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *Int. Conf. on Machine Learning*, 2016.

[19] J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models. In *Conf. on Neural Information Processing Systems*, 2020.

[20] Y.-N. Hung, I.-T. Chiang, Y.-A. Chen, and Y.-H. Yang. Musical Composition Style Transfer via Disentangled Timbre Representations. In *Int. Joint Conf. on Artificial Intelligence*, 2019.

[21] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi. Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms. *arXiv preprint arXiv:1812.08466*, 2018.

[22] H. Kim and A. Mnih. Disentangling by Factorising. In *Int. Conf. on Machine Learning*, 2018.

[23] J. W. Kim, R. Bittner, A. Kumar, and J. P. Bello. Neural Music Synthesis for Flexible Timbre Control. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2018.

[24] J. W. Kim, J. Salamon, P. Li, and J. P. Bello. Crepe: A Convolutional Representation for Pitch Estimation. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2018.

[25] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Int. Conf. on Learning Representations*, 2015.

[26] D. P. Kingma and M. Welling. Auto-encoding Variational Bayes. In *Int. Conf. on Learning Representations*, 2014.

[27] J. H. Lee, H.-S. Choi, and K. Lee. Audio Query-based Music Source Separation. In *Int. Soc. for Music Information Retrieval*, 2019.

[28] L. Lin, Q. Kong, J. Jiang, and G. Xia. A Unified Model for Zero-shot Music Source Separation, Transcription and Synthesis. In *Int. Soc. for Music Information Retrieval*, 2021.

[29] H. Liu, Y. Yuan, X. Liu, X. Mei, Q. Kong, Q. Tian, Y. Wang, W. Wang, Y. Wang, and M. D. Plumbley. AudioLDM 2: Learning Holistic Audio Generation With Self-Supervised Pretraining. *Trans. on Audio, Speech, and Language Processing*, 2024.

[30] X. Liu, D. Chin, Y. Huang, and G. Xia. Learning Interpretable Low-dimensional Representation via Physical Symmetry. In *Conf. on Neural Information Processing Systems*, 2023.

[31] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. Object-Centric Learning with Slot Attention. In *Conf. on Neural Information Processing Systems*, 2020.

[32] I. Loshchilov and F. Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Int. Conf. on Learning Representations*, 2017.

[33] Y.-J. Luo, K. Agres, and D. Herremans. Learning Disentengled Representations of Timbre and Pitch for Musical Instrument Sounds Using Gaussian Mixture Variational Autoencoders. In *Int. Soc. for Music Information Retrieval*, 2019.

[34] Y.-J. Luo, K. W. Cheuk, T. Nakano, M. Goto, and D. Herremans. Unsupervised Disentanglement of Pitch and Timbre for Isolated Musical Instrument Sounds. In *Int. Soc. for Music Information Retrieval*, 2020.

[35] Y.-J. Luo, S. Ewert, and S. Dixon. Towards Robust Unsupervised Disentanglement of Sequential Data — A Case Study Using Music Audio. In *Int. Joint Conf. on Artificial Intelligence*, 2022.

[36] Y.-J. Luo, S. Ewert, and S. Dixon. Unsupervised Pitch-Timbre Disentanglement of Musical Instruments Using a Jacobian Disentangled Sequential Autoencoder. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2024.

[37] M. Mancusi, E. Postolache, G. Mariani, M. Fumero, A. Santilli, L. Cosmo, and E. Rodolà. Unsupervised Source Separation via Bayesian Inference in the Latent Domain. *arXiv preprint arXiv:2110.05313*, 2021.

[38] E. Manilow, C. Hawthorne, C.-Z. A. Huang, B. Pardo, and J. Engel. Improving Source Separation by Explicitly Modeling Dependencies Between Sources. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2022.

[39] G. Mariani, I. Tallini, E. Postolache, M. Mancusi, L. Cosmo, and E. Rodolà. Multi-Source Diffusion Models for Simultaneous Music Generation and Separation. In *Int. Conf. on Learning Representations*, 2024.

[40] J. D. Parker, J. Spijkervet, K. Kosta, F. Yesiler, B. Kuznetsov, J.-C. Wang, M. Avent, J. Chen, and D. Le. StemGen: A Music Generation Model That Listens. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2024.

[41] M. Pasini, S. Lattner, and G. Fazekas. Self-Supervised Music Source Separation Using Vector-Quantized Source Category Estimates. *arXiv preprint arXiv:2311.13058*, 2023.

[42] W. Peebles and S. Xie. Scalable Diffusion Models with Transformers. In *Int. Conf. on Computer Vision*, 2023.

[43] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. FiLM: Visual Reasoning With a General Conditioning Layer. In *AAAI Conf. on Artificial Intelligence*, 2018.

[44] P. Reddy, S. Wisdom, K. Greff, J. R. Hershey, and T. Kipf. Audioslots: A Slot-Centric Generative Model For Audio Separation. In *Int. Conf. on Acoustics, Speech, and Signal Processing Workshops*, 2023.

[45] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *Int. Conf. on Computer Vision and Pattern Recognition*, 2022.

[46] G. Singh, Y. Kim, and S. Ahn. Neural Systematic Binder. In *Int. Conf. on Learning Representations*, 2022.

[47] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. In *Int. Conf. on Machine Learning*, 2015.

[48] K. Tanaka, Y. Bando, K. Yoshii, and S. Morishima. Unsupervised Disentanglement of Timbral, Pitch, and Variation Features From Musical Instrument Sounds With Random Perturbation. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2022.

[49] K. Tanaka, R. Nishikimi, Y. Bando, K. Yoshii, and S. Morishima. Pitch-Timbre Disentanglement Of Musical Instrument Sounds Based On Vae-Based Metric Learning. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2021.

[50] M. Tschannen, O. Bachem, and M. Lucic. Recent Advances in Autoencoder-Based Representation Learning. *arXiv preprint arXiv:1812.05069*, 2018.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is All you Need. In *Conf. on Neural Information Processing Systems*, 2017.

[52] Y. Wang, D. Stoller, R. Bittner, and J. Bello. Few-Shot Musical Source Separation. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2022.

[53] Y. Wu, J. Gardner, E. Manilow, I. Simon, C. Hawthorne, and J. Engel. The Chamber Ensemble Generator: Limitless High-Quality MIR Data via Generative Modeling. *arXiv preprint arXiv:2209.14458*, 2022.

[54] Y. Wu, Y. He, X. Liu, Y. Wang, and R. B. Dannenberg. Transplayer: Timbre Style Transfer with Flexible Timbre Control. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 2023.

[55] Y. Wu, Z. Wang, B. Raj, and G. Xia. Emergent Interpretable Symbols and Content-Style Disentanglement via Variance-Invariance Constraints. *arXiv preprint arXiv:2407.03824*, 2024.

[56] Y.-F. Wu, M. Lee, and S. Ahn. Neural Language of Thought Models. In *Int. Conf. on Learning Representations*, 2023.

[57] Z. Wu, Y. Rubanova, R. Kabra, D. A. Hudson, I. Gilitschenski, Y. Aytar, S. van Steenkiste, K. R. Allen, and T. Kipf. Neural Assets: 3D-Aware Multi-Object Scene Synthesis with Image Diffusion Models. *arXiv preprint arXiv:2406.09292*, 2024.

[58] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In *Int. Conf. on Machine Learning*, 2021.

A Derivation of the ELBO

For a multi-instrument mixture containing N_s sources, we define its corresponding sets of the pitch annotations, the timbre latents, and the pitch latents as $\mathcal{Y} = \{y^{(i)}\}_{i=1}^{N_s}$, $\mathcal{T} = \{\tau^{(i)}\}_{i=1}^{N_s}$, and $\mathcal{V} = \{\nu^{(i)}\}_{i=1}^{N_s}$, respectively. An instrument can be represented by a source-level representation $s^{(i)} = \mathrm{f}_{\theta_s}(\tau^{(i)}, \nu^{(i)})$, where $\mathrm{f}_{\theta_s}(\cdot)$ is a deterministic function, the diamond node that precedes x_m illustrated on the left side of Fig. 1b. Similarly, the set of source-level representations is denoted as $\mathcal{S} = \{s^{(i)}\}_{i=1}^{N_s}$ and can be expressed in terms of $\mathcal{T}$ and $\mathcal{V}$ by $\mathrm{f}_{\theta_s}(\mathcal{T}, \mathcal{V})$.

In the following derivation of the objective function $\mathcal{L}_{\text{ELBO}}$ (5), we begin with the defined set notations, expand and simplify the equations with the factorised form of the posteriors, and recover $\mathcal{L}_{\text{ELBO}}$ based on the deterministic posterior over the pitch latent.

$$\log p(x_m|\mathcal{Y}) \geq \mathcal{L}_{\text{ELBO}} \tag{6}$$

$$= \mathbb{E}_{q_{\phi_\tau}(\mathcal{T}|x_m,\mathcal{X}_q)q_{\phi_\nu}(\mathcal{V}|x_m,\mathcal{X}_q)}\big[\log p_{\theta_m}(x_m|\mathcal{S} = \mathrm{f}_{\theta_s}(\mathcal{T}, \mathcal{V}))\big] \tag{7}$$

$$- \mathcal{D}_{\text{KL}}\big(q_{\phi_\tau}(\mathcal{T}|x_m, \mathcal{X}_q)\|p(\mathcal{T})\big) - \mathcal{D}_{\text{KL}}\big(q_{\phi_\nu}(\mathcal{V}|x_m, \mathcal{X}_q)\|p_{\theta_\nu}(\mathcal{V}|\mathcal{Y})\big) \tag{8}$$

$$= \mathbb{E}_{\prod_{i=1}^{N_s} q_{\phi_\tau}(\tau^{(i)}|x_m,x_q^{(i)})q_{\phi_\nu}(\nu^{(i)}|x_m,x_q^{(i)})}\big[\log p_{\theta_m}(x_m|\mathcal{S} = \{\mathrm{f}_{\theta_s}(\tau^{(i)}, \nu^{(i)})\}_{i=1}^{N_s})\big] \tag{9}$$

$$- \int \prod_{i=1}^{N_s} q_{\phi_\tau}(\tau^{(i)}|x_m, x_q^{(i)}) \log \frac{\prod_{i=1}^{N_s} q_{\phi_\tau}(\tau^{(i)}|x_m, x_q^{(i)})}{\prod_{i=1}^{N_s} p(\tau^{(i)})} d\tau^{(1)} \ldots d\tau^{(N_s)} \tag{10}$$

$$- \int \prod_{i=1}^{N_s} q_{\phi_\nu}(\nu^{(i)}|x_m, x_q^{(i)}) \log \frac{\prod_{i=1}^{N_s} q_{\phi_\nu}(\nu^{(i)}|x_m, x_q^{(i)})}{\prod_{i=1}^{N_s} p_{\theta_\nu}(\nu^{(i)}|y^{(i)})} d\nu^{(1)} \ldots d\nu^{(N_s)} \tag{11}$$

$$= \mathbb{E}_{\prod_{i=1}^{N_s} q_{\phi_\tau}(\tau^{(i)}|x_m,x_q^{(i)})q_{\phi_\nu}(\nu^{(i)}|x_m,x_q^{(i)})}\big[\log p_{\theta_m}(x_m|\mathcal{S} = \{\mathrm{f}_{\theta_s}(\tau^{(i)}, \nu^{(i)})\}_{i=1}^{N_s})\big] \tag{12}$$

$$- \int \prod_{i=1}^{N_s} q_{\phi_\tau}(\tau^{(i)}|x_m, x_q^{(i)}) \sum_{i=1}^{N_s} \log \frac{q_{\phi_\tau}(\tau^{(i)}|x_m, x_q^{(i)})}{p(\tau^{(i)})} d\tau^{(1)} \ldots d\tau^{(N_s)} \tag{13}$$

$$- \int \prod_{i=1}^{N_s} q_{\phi_\nu}(\nu^{(i)}|x_m, x_q^{(i)}) \sum_{i=1}^{N_s} \log \frac{q_{\phi_\nu}(\nu^{(i)}|x_m, x_q^{(i)})}{p_{\theta_\nu}(\nu^{(i)}|y^{(i)})} d\nu^{(1)} \ldots d\nu^{(N_s)} \tag{14}$$

$$= \mathbb{E}_{\prod_{i=1}^{N_s} q_{\phi_\tau}(\tau^{(i)}|x_m,x_q^{(i)})q_{\phi_\nu}(\nu^{(i)}|x_m,x_q^{(i)})}\big[\log p_{\theta_m}(x_m|\mathcal{S} = \{\mathrm{f}_{\theta_s}(\tau^{(i)}, \nu^{(i)})\}_{i=1}^{N_s})\big] \tag{15}$$

$$- \sum_{i=1}^{N_s} \int q_{\phi_\tau}(\tau^{(i)}|x_m, x_q^{(i)}) \log \frac{q_{\phi_\tau}(\tau^{(i)}|x_m, x_q^{(i)})}{p(\tau^{(i)})} d\tau^{(i)} \tag{16}$$

$$- \sum_{i=1}^{N_s} \int q_{\phi_\nu}(\nu^{(i)}|x_m, x_q^{(i)}) \log \frac{q_{\phi_\nu}(\nu^{(i)}|x_m, x_q^{(i)})}{p_{\theta_\nu}(\nu^{(i)}|y^{(i)})} d\nu^{(i)} \tag{17}$$

$$= \mathbb{E}_{\prod_{i=1}^{N_s} q_{\phi_\tau}(\tau^{(i)}|x_m,x_q^{(i)})\delta(\nu^{(i)} - \mathrm{f}_{\phi_\nu}(\hat{y}_{\text{bin}}^{(i)}))}\big[\log p_{\theta_m}(x_m|\mathcal{S} = \{\mathrm{f}_{\theta_s}(\tau^{(i)}, \nu^{(i)})\}_{i=1}^{N_s})\big] \tag{18}$$

$$- \sum_{i=1}^{N_s} \mathcal{D}_{\text{KL}}\big(q_{\phi_\tau}(\tau^{(i)}|x_m, x_q^{(i)})\|p(\tau^{(i)})\big) \tag{19}$$

$$+ H(\delta(\nu^{(i)} - \mathrm{f}_{\phi_\nu}(\hat{y}_{\text{bin}}^{(i)}))) + \sum_{i=1}^{N_s} \int \delta(\nu^{(i)} - \mathrm{f}_{\phi_\nu}(\hat{y}_{\text{bin}}^{(i)})) \log p_{\theta_\nu}(\nu^{(i)}|y^{(i)})) d\nu^{(i)} \tag{20}$$

$$= \mathbb{E}_{\prod_{i=1}^{N_s} q_{\phi_\tau}(\tau^{(i)}|x_m,x_q^{(i)})}\big[\log p_{\theta_m}(x_m|\mathcal{S} = \{\mathrm{f}_{\theta_s}(\tau^{(i)}, \hat{\nu}^{(i)})\}_{i=1}^{N_s})\big] \tag{21}$$

$$- \sum_{i=1}^{N_s} \mathcal{D}_{\text{KL}}\big(q_{\phi_\tau}(\tau^{(i)}|x_m, x_q^{(i)})\|p(\tau^{(i)})\big) + \sum_{i=1}^{N_s} \log p_{\theta_\nu}(\hat{\nu}^{(i)}|y^{(i)}). \tag{22}$$

Starting with the standard form of the evidence lower bound (ELBO), we first assume that the posteriors over $\mathcal{T}$ and $\mathcal{V}$ are factorised in term (7), and thus the two terms of the Kullback–Leibler divergence (KLD) corresponding to $\mathcal{T}$ and $\mathcal{V}$ are written as in term (8). In terms (9) to (11), we further assume and factorise each posterior into a product over N_s sources. Note that the two KLD terms are expanded by the definition of KLD. Based on the configuration of a deterministic posterior over $\nu^{(i)}$, we rewrite $q_{\phi_\nu}(\nu^{(i)}|x_m, x_q^{(i)})$ as the Dirac delta function in terms (18) and (20). Finally, by replacing $\nu^{(i)}$ with $\hat{\nu}^{(i)} = \mathrm{f}_{\phi_\nu}(\hat{y}_{\text{bin}}^{(i)})$, we recover $\mathcal{L}_{\text{ELBO}}$ (5).

B Implementation Details

B.1 The Simple Case Study Using an Autoencoder

Mixture Reconstruction $p_{\theta_m}(x_m|\mathcal{S})$ in $\mathcal{L}_{\text{ELBO}}$ (5) is a Gaussian likelihood parameterised by a decoder $\mathrm{D}_{\theta_m}(\cdot)$, which can be a permutation invariant function such as a transformer [51] without positional embeddings that outputs the Gaussian mean, reconstructing a mixture given a set of source-level representations $\mathcal{S}$. We opt for the simple implementation for this case study and discuss a transformer in Section 4.2 and Appendix B.2.

In particular, we slightly deviate from $\mathcal{L}_{\text{ELBO}}$ (5) and reconstruct x_m using e_m as illustrated in Fig. 1a, instead of $\mathcal{S}$, whereby the likelihood becomes $p_{\theta_m}(x_m|e_m)$, and we maximise an additional likelihood:

$$\mathbb{E}_{\prod_i q_{\phi_\tau}(\tau^{(i)})}\big[\log p(e_m|\{\tau^{(i)}, \hat{\nu}^{(i)}\}_{i=1}^{N_s}\big]. \tag{23}$$

Recall that $\mathcal{S} = \{s^{(i)}\}_{i=1}^{N_s}$; $s^{(i)} = \mathrm{f}_{\theta_s}(\tau^{(i)}, \hat{\nu}^{(i)})$, and thus the original conditions $\mathcal{S}$ and x_m are "bridged" by e_m via the objective (23) and the fact that e_m is used to reconstruct x_m. We let the likelihood $p(e_m|\{\tau^{(i)}, \hat{\nu}^{(i)}\}_{i=1}^{N_s}) = \mathcal{N}(e_m; \sum_{i=1}^{N_s} s^{(i)}, \sigma_m^2 \mathrm{I})$ and $\sigma_m = 0.25$ is a hyperparameter. Intuitively speaking, by maximising the objective (23), the summation of the source-level latents $s_{\text{sum}} = \sum_i s^{(i)}$ and e_m are pulled together, with the distance measured in terms of a mean square error weighed by σ_m^{-2}. We can also interpret the likelihood $p(e_m|\{\tau^{(i)}, \hat{\nu}^{(i)}\}_{i=1}^{N_s})$ as a conditional prior that constrains e_m.

It is important to note that during evaluation, we instead use s_{sum} to reconstruct the mixture x_m or render a novel one by manipulating $\{s^{(i)}\}_{i=1}^{N_s}$ before the summation, indicated by the dashed blue arrow in Fig. 1a. The underlying assumption is that both e_m and s_{sum} can reconstruct x_m comparably well as a result of maximising the objective (23) (so that e_m and s_{sum} are pulled together) and reconstructing x_m by e_m during training.

This approach of mixture reconstruction deviates from Eq. (1) and avoids implementing a (potentially expensive and complicated) permutation invariant decoder. Although we could have parameterised the original likelihood as $p_{\theta_m}(x_m|\mathcal{S}) = \mathcal{N}(x_m; \mathrm{D}_{\theta_m}(\sum_{i=1}^{N_s} s^{(i)}), \sigma_m^2 \mathrm{I})$, which does not require introducing additional loss terms, we opted for our approach and introduced the objective (23) which imposes a linearity between e_m and s_{sum} in the latent space [37].

This linearity could potentially allow for manipulating specific sources without extracting the entire set of source-level representations. To elaborate, we can manipulate a specific source i and sample a novel mixture by first computing a residual $r = e_m - s^{(i)}$, where e_m is computed from the given mixture and $s^{(i)}$ is inferred conditioned on the mixture and a single query. Then, we manipulate $s^{(i)}$ to obtain $\hat{s}^{(i)}$ that contains the desired pitch and timbre, and finally use $\hat{e}_m = r + \hat{s}^{(i)}$ to render a novel mixture. We only require a single query to manipulate the attributes of a target source and render a new mixture, unlike the alternative approach, where N_s queries are required even if we are only interested in manipulating a single source. Studying the linearity is left for future work.

Pitch and Timbre Combination We employ FiLM [43, 23] and derive the source-level representation as $s^{(i)} = \mathrm{f}_{\theta_s}(\tau^{(i)}, \nu^{(i)}) = \alpha_{\theta_s}(\tau^{(i)}) \odot \nu^{(i)} + \beta_{\theta_s}(\tau^{(i)})$, where $\nu^{(i)}$ is scaled and shifted element-wise by factors $\alpha_{\theta_s}(\cdot)$ and $\beta_{\theta_s}(\cdot)$ determined by $\tau^{(i)}$.

Dataset The authors of MusicSlots [15] compiled a synthetic audio dataset using 3,131 unique chords from JSB Chorales [5], rendered by sound fonts of piano, violin, and flute via FluidSynth.

Given a chord, each composite note is synthesised to an audio waveform at 16kHz with a sound font randomly sampled with replacement, whereby a sound font i can play multiple notes in a chord. These notes together define a source $x_s^{(i)}$ whose pitch annotation is a mutli-hot vector $y^{(i)} \in \{0, 1\}^{N_p}$ and $N_p = 52$. The note waveforms are summed to form the chord waveform which defines a mixture.

There are 28,179 samples of mixtures split into the train, validation, and test sets with a ratio of 70/20/10. The waveforms are converted into mel spectrograms using 128 mel-filter bands, a window size of 1,024, and a hop length of 512. We use a segment of 320ms that is cropped from the sustain phase of each sample, which amounts to 10 spectral frames per data for both training and evaluation.

Table 3: The $\mathrm{Conv1D}$ layers used to implement $\mathrm{E}_{\phi_m}(\cdot)$ and $\mathrm{E}_{\phi_q}(\cdot)$ for the simple model in Section 4.1.

Inp. channel	**Out. channel**	**Kernel size**	**Stride**	**Padding**	**Normalization**	**Activation**
128	768	3	1	0	Layer	ReLU
768	768	3	1	1	Layer	ReLU
768	768	4	2	1	Layer	ReLU
768	768	3	1	1	Layer	ReLU
768	768	3	1	1	Layer	ReLU
768	64	1	1	1	None	None

Table 4: The three modules of the pitch encoder for the simple model in Section 4.1: The transcriber $\mathrm{E}_{\phi_\nu}(\cdot)$, the SB layer, and the projector $\mathrm{f}_{\phi_\nu}(\cdot)$.

Module	**Input size**	**Output size**	**Normalization**	**Activation**
$\mathrm{E}_{\phi_\nu}(\cdot)$	128	256	Layer	ReLU
	256	256	Layer	ReLU
	256	256	Layer	ReLU
	256	256	Layer	ReLU
	256	256	Layer	ReLU
	256	256	Layer	ReLU
	256	52	None	Sigmoid
Stochastic Binarisation (SB)				
$\mathrm{f}_{\phi_\nu}(\cdot)$	52	64	None	ReLU
	64	64	None	ReLU
	64	64	None	None

Architecture The mixture encoder $\mathrm{E}_{\phi_m}(\cdot)$ and the query encoder $\mathrm{E}_{\phi_q}(\cdot)$, mentioned in Section 3.1, share an architecture which is a stack of $\mathrm{Conv1D}$ layers that take as input the mel spectrograms of x_m and $x_q^{(i)}$ and output $e_m, e_q^{(i)} \in \mathbb{R}^{64}$, respectively. Table 3 outlines the architecture. The encoders are followed by a mean pooling along the temporal dimension such that a mel spectrogram of $\mathbb{R}^{128\times 10}$ is projected to $e_m, e_q^{(i)} \in \mathbb{R}^{64}$. The two are concatenated as a 128-dimensional vector used to extract the pitch and timbre latents.

Given the concatenation of e_m and $e_q^{(i)}$, the pitch encoder first transcribes $\hat{y}^{(i)} \in \mathbb{R}^{N_p}$ using $\mathrm{E}_{\phi_\nu} : \mathbb{R}^{128} \to \mathbb{R}^{N_p}$, where $N_p = 52$. $\nu^{(i)} \in \mathbb{R}^{64}$ is then extracted by applying $\mathrm{f}_{\phi_\nu} : \mathbb{R}^{N_p} \to \mathbb{R}^{64}$ to $\hat{y}_{\text{bin}}^{(i)}$. Table 4 shows the complete architecture. The timbre encoder shares the same architecture as $\mathrm{E}_{\phi_\nu}(\cdot)$, except for that the last layer is replaced by a Gaussian parameterisation layer which consists of two linear layers that project the 256-dimensional hidden state to the mean and log-variance of the posterior, from which the timbre latent $\tau^{(i)} \in \mathbb{R}^{64}$ is sampled.

The FiLM layer $\mathrm{f}_{\theta_s}(\cdot)$ obtains the source-level latent $s^{(i)}$ by combining the pitch and the timbre latent. In particular, the timbre latent is linearly transformed to compute the scaling and shifting factors, and the pitch latent is modulated by the factors.

To reconstruct the mel spectrograms, the source-level latent $s^{(i)}$ is then temporally broadcast to match the number of time frames 10 of the mel spectrograms. A two-layer bi-directrional gated recurrent unit (GRU) then transforms the broadcast $s^{(i)}$ to an output of $\mathbb{R}^{128\times 10}$ which is then processed by a linear layer to reconstruct the input mel spectrograms.

Pitch Priors We study pitch priors at different levels of capacity. The first follows Eq. (1) and defines $p_{\theta_\nu}(\nu^{(i)}|y^{(i)}) = \mathcal{N}(\nu^{(i)}; \mu_{\theta_\nu}^{\text{fac}}(y^{(i)}), \sigma_\nu^2 \mathrm{I})$, a Gaussian parameterised by a neural network $\mu_{\theta_\nu}^{\text{fac}}(\cdot)$ given the ground-truth pitch $y^{(i)}$. $\sigma_\nu = 0.5$ is a hyperparameter.

The prior network first projects $y^{(i)} \in \mathbb{R}^{N_p}$ by reusing $\mathrm{f}_{\phi_\nu} : \mathbb{R}^{N_p} \to \mathbb{R}^{64}$. The rest of the network shares the same architecture as $\mathrm{E}_{\phi_\nu}(\cdot)$ in Table 4, except for that the input size is 64 instead of

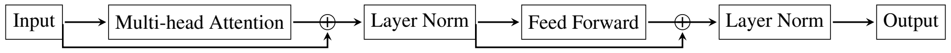


Figure 4: A regular post-norm transformer block.

128 and the last layer is replaced by two linear layers representing the mean and log-variance of a 64-dimensional Gaussian distribution.

We also consider a more expressive counterpart to capture the source interaction: $p_{\theta_\nu}(\nu^{(i)}|\mathcal{Y}_{\setminus i}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\big(\nu^{(i)}; \mu^{\text{rich}}_{\theta_\nu,k}(\mathcal{Y}_{\setminus i}), \sigma^2_\nu \mathrm{I}\big)$, where $\mathcal{Y}_{\setminus i}$ denotes a set of pitch annotations excluding that of the i-th source, and $\mu^{\text{rich}}_{\theta_\nu,k}(\cdot)$ specifies the mean of the k-th component in a Gaussian mixture. The rationale is that $\hat{y}^{(i)}$ is conditionally dependent on pitch of other sources in a mixture to confer musical harmony.

The expressive prior network first transforms each element from the set of pitch annotations $\mathcal{Y}_{\setminus i}$ by $\mathrm{f}_{\phi_\nu}(\cdot)$. To handle a set of inputs, we implement a transformer. The architecture consists of three blocks of a regular post-norm transformer, as shown in Fig. 4. We use a four-head attention and an embedding size of 64. The feed-forward block is a two-layer MLP with a ReLU and a size of 64.

The elements in the set $\mathcal{Y}_{\setminus i}$ are treated as a sequence of tokens and fed to the transformer without adding positional encodings to preserve permutation invariance. A max pooling is applied to collapse the $N_s - 1$ tokens in the set $\mathcal{Y}_{\setminus i}$ to an output hidden state, followed by one of the K Gaussian parameterisation layers to map the hidden state to the mean and log-variance of a Gaussian component. The Gaussian layers consists of $2 \times K$ linear layers to specify the mean and logvariance of K components in the Gaussian mixture.

Optimisation We use Adam [25] and a batch size of 32, a learning rate of 0.0004, and a gradient clipping value of 0.5. Training is terminated if the loss function stops improving on the validation set for 260k steps.

B.2 The Latent Diffusion Model

We also implement DisMix using an LDM framework [45], where the likelihood $p_{\theta_m}(x_m|\mathcal{S})$ is implemented by a diffusion transformer (DiT) [42] that directly reconstructs a mixture given a set of source-level latents, and no additional objective such as (23) is introduced.

Data Representation The LDM framework [45] improves the compute efficiency of diffusion models (DMs) [47, 19] by first projecting data to a low-dimensional latent space. We leverage the pre-trained VAE from AudioLDM2 [29], which is trained using multiple music and audio datasets, to extract $z_s^{(i)} = \mathrm{E}_{\text{vae}}(x_s^{(i)})$, where $\mathrm{E}_{\text{vae}}(\cdot)$ is the VAE encoder. The reconstruction is by $x_s^{(i)} = \mathrm{D}_{\text{vae}}(z_s^{(i)})$, where $\mathrm{D}_{\text{vae}}(\cdot)$ is the pre-trained VAE decoder.

Latent Diffusion Models To facilitate understanding of our DiT implementation, we briefly review DMs. LDMs operate DMs in a latent space and sample z_0, a latent feature of data, from a Markov chain: $p(z_T)\prod_{t=1}^{T} p_\theta(z_{t-1}|z_t)$, with $p(z_T) = \mathcal{N}(z_T; 0,\mathrm{I})$ and $p_\theta(z_{t-1}|z_t) = \mathcal{N}(z_{t-1}; \mu_\theta(z_t,t), \Sigma_\theta(z_t,t))$ parameterised by neural networks $\mu_\theta(\cdot)$ and $\Sigma_\theta(\cdot)$. In other words, a DM samples z_0 through an iterative process of T de-noising steps (otherwise known as the reverse process), starting from the standard Gaussian distribution.

The posterior is a linear Gaussian $q(z_t|z_{t-1}) = \mathcal{N}(z_t; \sqrt{\alpha_t} z_{t-1}, (1-\alpha_t)\mathrm{I})$, where α_t is a hyperparameter evolving over the diffusing step t, specified by a noise schedule. Given that the posterior is known and fixed, DDPM [19] employs specific forms of $\mu_\theta(\cdot)$ and $\Sigma_\theta(\cdot)$ based on the posterior and simplifies the training to essentially minimising $\|\mathrm{f}_\theta(z_t,t) - z_0\|_2^2$, which boils down to training a decoder $\mathrm{f}_\theta(\cdot)$ to predict the clean z_0 given its corrupted counterpart z_t.

Mixture Reconstruction As illustrated by the red arrows in Fig. 1a, the mixture x_m is reconstructed by summing the constituent sources $\{x_s^{(i)}\}_{i=1}^{N_s}$ which are recovered by $\mathrm{D}_{\text{vae}}(\cdot)$ from $\{z_s^{(i)}\}_{i=1}^{N_s}$. To this end, we implement a DiT to reconstruct $\{z_s^{(i)}\}_{i=1}^{N_s}$ given $\mathcal{S} = \{s^{(i)}\}_{i=1}^{N_s}$. That is, the likelihood $p_{\theta_m}(x_m|\mathcal{S})$ in $\mathcal{L}_{\text{ELBO}}$ (5) is re-written as follows:

$$p_{\theta_m}(\{z_{s,0:T}^{(i)}\}_{i=1}^{N_s}|\mathcal{S}) = p(\{z_{s,T}^{(i)}\}_{i=1}^{N_s}) \prod_{t=1}^{T} p_{\theta_m}(\{z_{s,t-1}^{(i)}\}_{i=1}^{N_s}|\{z_{s,t}^{(i)}\}_{i=1}^{N_s}, \mathcal{S}). \tag{24}$$

Table 5: The $\mathrm{Conv1D}$ layers used to implement the timbre encoder for the LDM in Section 4.2. The last row refers to a Gaussian layer where the 256-dimensional output is split to represent the mean and log-variance of the posterior. The number in the parenthesis indicates the number of groups divided for normalisation.

Inp. channel	Out. channel	Kernel size	Stride	Padding	Normalization	Activation
128	128	5	2	0	Group(1)	ReLU
128	128	5	2	0	Group(1)	ReLU
128	128	5	2	0	Group(1)	ReLU
128	256	1	1	0	None	None

$z_{s,t}^{(i)}$ denotes the noised latent feature $z_s^{(i)} = \mathrm{E}_{\mathrm{vae}}(x_s^{(i)})$ at the diffusing step t and $z_{s,0}^{(i)} = z_s^{(i)}$. In other words, the DiT is trained to predict the clean $z_s^{(i)}$ given its noisy counterpart $z_{s,t}^{(i)}$ and the source-level pitch and timbre representations $s^{(i)}$. Note that we condition the DiT on a set of N_s noisy features and ask it to predict the corresponding set of clean features in one pass of the model during training (while there are T iterations during evaluation). Therefore, the model is able to capture interdependencies across the N_s sources. We have shown that this setup (M1 with "Set" mode) outperforms its single source-conditioned counterpart (M0 with "Single" mode) in Table 2.

Dataset The CococChorale dataset [53] provides realistic generative music in the style of Bach Chorales [5], featuring 13 orchestral instruments played at the pitch ranges of a standard four-part chorale (i.e., Soprano, Alto, Tensor, Bass, or SATB), and $N_s = 4$.

We use the random ensemble subset of 60,000 four-part chorales, totalling 350 hours of audio. Each part of an example is played by an instrument randomly sampled from a pool of instruments belonging to the part. The pitch annotation $y^{(i)} \in \{0,1\}^{N_p \times T_p}$ is a time sequence of one-hot vectors, where $N_p = 129$. The dataset follows a split ratio of 80/10/10 for the training, validation, and test sets.

Each audio file is sampled at 16kHz and is converted into mel spectrograms using 64 mel-filter bands, a window size of 1,024, and a hop length of 160. We randomly sample a four-second segment for each example and $T_p = 400$.

Encoders We extract mel spectrograms of $\mathbb{R}^{64\times 400}$, where the dimensions correspond to the frequency and time axes, respectively. The mixture encoder $\mathrm{E}_{\phi_m}(\cdot)$ processes the mel spectrograms of x_m and outputs $e_m \in \mathbb{R}^{8\times 100\times 16}$, with the dimensions corresponding to the channels, time frames, and feature size, respectively. We use and freeze $\mathrm{E}_{\mathrm{vae}}(\cdot)$, the pre-trained encoder of the VAE in AudioLDM2 [29] as the mixture encoder. Similarly, we reuse the architecture of $\mathrm{E}_{\mathrm{vae}}(\cdot)$ for the query encoder $\mathrm{E}_{\phi_q}(\cdot)$ but train it from scratch and append a temporal pooling layer, which outputs $e_q^{(i)} \in \mathbb{R}^{8\times 1\times 16}$. To combine the information of $e_q^{(i)}$ and e_m for extracting the pitch and timbre latents, $e_q^{(i)}$ is broadcast along the time dimension and concatenated with e_m along the feature dimension. The outcome is then transformed back to the original feature dimension of 16 by a 1×1 $\mathrm{Conv2D}$ filter and passed for pitch and timbre extraction.

Given the concatenation of e_m and $e_q^{(i)}$, the pitch encoder first transcribes with $\mathrm{E}_{\phi_\nu} : \mathbb{R}^{8\times 100\times 16} \rightarrow \mathbb{R}^{129\times 400}$ whose architecture is based on the decoder $\mathrm{D}_{\mathrm{vae}}(\cdot)$ from the pre-trained VAE in AudioLDM2. We modify the output dimension from 64 to 129 to accommodate the number of classes of pitch and train $\mathrm{E}_{\phi_\nu}(\cdot)$ from scratch. The output of $\mathrm{E}_{\phi_\nu}(\cdot)$ is then binarised by the SB layer and passed through $\mathrm{f}_{\phi_\nu} : \mathbb{R}^{129\times 400} \rightarrow \mathbb{R}^{8\times 100\times 16}$ to derive the pitch latent $\nu^{(i)}$, where $\mathrm{f}_{\phi_\nu}(\cdot)$ is based on the architecture of $\mathrm{E}_{\mathrm{vae}}(\cdot)$ and is trained from scratch.

The timbre encoder converts an input of $\mathbb{R}^{8\times 100\times 16}$, or the concatenation of e_m and $e_q^{(i)}$, to the timbre latent $\tau^{(i)} \in \mathbb{R}^{8\times 1\times 16}$. Table 5 specifies the architecture which is followed by a temporal pooling. The timbre latent is then broadcast over the time axis and concatenated with the pitch latent to construct the source-level latent $s^{(i)} \in \mathbb{R}^{8\times 100\times 32}$.

Partition DiTs are transformers that take as inputs sequences of patches [42] . We describe how input sequences are constructed from $\{z_s^{(i)}\}_{i=1}^{N_s}$ and $\{s^{(i)}\}_{i=1}^{N_s}$. We first apply a sinusoidal positional embedding to $z_s^{(i)}$ to preserve the temporal order and partition it by $\mathrm{Par}(z_s^{(i)}) : \mathbb{R}^{T_z\times(D_z\times C)} \rightarrow$

$\mathbb{R}^{L \times D'_z}$, where $T_z = 100$, $D_z = 16$, $C = 8$, and $L = 25$ are the numbers of time frames, feature dimensions, channels, and patches, respectively. That is, we partition along the time axis and flatten each patch whose size becomes $D'_z = \frac{T_z}{L} \times D_z \times C$. We repeat the process for all N_s elements in $\{z_s^{(i)}\}_{i=1}^{N_s}$. The outcome is a feature $z_m \in \mathbb{R}^{(N_s \times L) \times D'_z}$, where we define z_m as the aggregated representation of $\{z_s^{(i)}\}_{i=1}^{N_s}$ as the result of the partition. We can consider z_m as a sequence of $N_s \times L$ patches, with the size of each patch being D'_z.

Similarly, we partition the set of source-level representations $\mathcal{S} = \{s^{(i)}\}_{i=1}^{N_s}$ and obtain a aggregated representation $s_c \in \mathbb{R}^{(N_s \times L) \times D'_s}$, where $D'_s = \frac{T_z}{L} \times D_s \times C$ and we define s_c as the result of the partition. $D_s = 32$ as we have concatenated the pitch and the timbre latents. Therefore, the aggregated representations, z_m and s_c, of $\{z_s^{(i)}\}_{i=1}^{N_s}$ and $\{s^{(i)}\}_{i=1}^{N_s}$, respectively, share common dimensions except for their patch sizes D'_z and D'_s.

At this point, we can re-write Eq. (24) as:

$$p_{\theta_m}(z_{m,0:T}|\mathcal{S}) = p(z_{m,T}) \prod_{t=1}^{T} p_{\theta_m}(z_{m,t-1}|z_{m,t}, s_c). \tag{25}$$

We do not add another positional embedding to z_m and s_c to ensure the permutation invariance with respect to the N_s sources.

Conditioning We describe how we condition the reconstruction of $z_{m,t-1}$ on $z_{m,t}$ and s_c (Eq. (25)). A transformer block consists of the multi-head self-attention mechanism and a feedforward network, as depicted in Fig. 4. Each of these modules is followed by a skip connection and a layer normalisation (LN) [51]. We replace the standard LN with its adaptive variant (adaLN) for the conditioning [42].

To inform the DiT about the level of noise used to corrupt $z_{m,t}$, an embedding of the diffusing step t is first added to s_c to form the condition. The condition is then used to regress the scaling and shifting factors of the adaLN layers. Then, passing $z_{m,t}$ through the transformer blocks modulates $z_{m,t}$ by the factors which carry the source-level pitch and timbre information provided by s_c.

Decoder The DiT consists of three regular transformer blocks illustrated in Fig. 4 and we set the number of heads to four for the multi-head self-attention mechanism. The embedding size is the same as the input size D'_z. As previously described, we replace the layer normalisation layers with its adaptive counterparts to condition the input $z_{m,t}$ with s_c.

Optimisation We use Adam [25] with a batch size of eight and a gradient clipping value of 0.5. The learning rate warms up linearly to 0.0001 over 308k steps and decreases following the cosine wave [32] for a maximum of 4,092k steps.

C Additional Results

C.1 The Simple Case Study Using an Autoencoder

Evaluation Protocol Given a mixture x_m and a set of queries $\{x_q^{(i)}\}_{i=1}^{N_s}$ corresponding to the N_s constituent instruments of the mixture, we first extract the timbre and pitch latents $\{\tau^{(i)}, \nu^{(i)}\}_{i=1}^{N_s}$. To evaluate disentanglement, we conduct a random permutation so that the pitch latent of the source i can be swapped for that of the source j, while the timbre remains unchanged, which yields a novel source-level latent $\hat{s}^{(i)} = \mathrm{f}_{\theta_s}(\tau^{(i)}, \nu^{(j)})$. Then, we render a novel source $\hat{x_s}^{(i)}$ by passing $\hat{s}^{(i)}$ to the decoder $\mathrm{D}_{\theta_m}(\cdot)$. Note that we render sources instead of mixtures by using $\hat{s}^{(i)}$ as the input instead of the summation of multiple source-level representations.

We pre-train a pitch and an instrument classifier using the training set. Successful disentanglement entails the pre-trained classifiers of pitch and instrument classify $\hat{x}_s^{(i)}$ as the pitch of $x_s^{(j)}$ (as it was swapped) and the instrument of $x_s^{(i)}$ (as it was preserved), respectively. The classification accuracy has been reported under "Disentanglement" in Table 1.

To see if the model can render novel mixtures, we first produce a new set $\{\hat{s}^{(i)}\}_{i=1}^{N_s}$ after the permutation and render a novel mixture $\hat{x}_m$ by passing $\mathrm{D}_{\theta_m}(\cdot)$ the summation $\hat{s}_{\text{sum}} = \sum_i \hat{s}^{(i)}$. To check whether the constituent attributes of $\hat{x}_m$ are indeed dictated by the manipulated $\{\hat{s}^{(i)}\}_{i=1}^{N_s}$, we

Table 6: Pitch classification accuracy (%) using different pitch priors.

	DisMix	+fac	+rich ($K = 1$)	+rich ($K = 10$)
Disentanglement	93.39	93.98	94.01	94.18
Mixture Rendering	90.69	91.15	91.39	92.04

once again extract its source-level representations with the original queries $\{x_q^{(i)}\}_{i=1}^{N_s}$ and reconstruct the sources, which are again fed to the pre-trained classifiers. The performance has been reported under "Mixture Rendering" in Table 1.

The instrument and pitch classifiers share the same architecture, except for their last layers corresponding to the three classes of instrument and and the 52 classes of pitch, respectively. Both of the classifiers start with the architecture outlined in Table 3 followed by a three-layer MLP. The input and output sizes of the first two layers of the three-layer MLP are 64 with the ReLU activation, while those of the last layer are 64 and the numbers of classes described above.

Pitch Priors Table 6 reports the performance of the model with different choices of pitch priors. DisMix is the proposed autoencoder without a pitch prior, discarding the second term from $\mathcal{L}_{\mathrm{ELBO}}$ (5). fac and rich denote the factorised and expressive priors described in Section B.1. We also experiment with the numbers of Gaussian components $K = 1$ and $K = 10$.

We can observe progressive improvement with a richer prior. The gain from fac to rich with $K = 1$ is rather marginal compared to the gain from fac to rich at $K = 10$. This is probably because the pitch distribution conditioned on contextual pitch information is multimodal and the use of a single Gaussian component would not provide much benefit compared to fac.

In addition to the improvement in disentanglement observed in Table 6, the rich variant of the pitch prior provides the possibility of sampling pitch latents conditioned on a context of pitch values [40, 39], which we leave for future work.

Examples The right side of Fig. 5 shows a example of attribute swapping within a mixture. The first row includes the queries for the three constituent instruments of the mixture. The second row shows the input mixture and the three underlying sources. The timbre characteristics are consistent across rows for the last three columns. In the third row, the leftmost column is the reconstructed mixture given s_{sum} and the rest are the reconstructed sources given $s^{(i)}$.

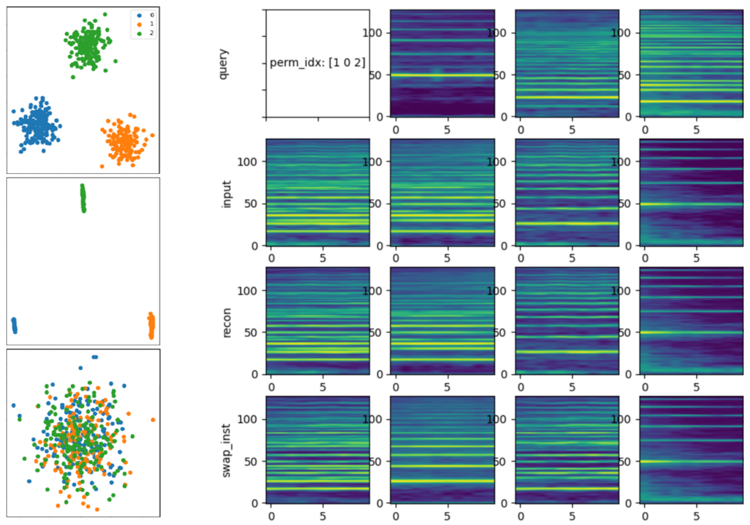


Figure 5: Additional results for the simple case study. *Left*: PCA of the timbre space. Top: DisMix, plot $\tau^{(i)}$. Mid and bottom: Remove $\mathcal{L}_{\mathrm{BT}}$, plot the mean of $q_{\phi_\tau}(\tau^{(i)})$ and the sampling, respectively. *Right*: Novel mixture rendering.

Table 7: The instrument classifier used to evaluate the disentanglement of the proposed LDM. The number in the parenthesis indicates the number of groups divided for normalisation.

Inp. channel	Out. channel	Kernel size	Stride	Padding	Normalization	Activation
64	64	5	2	0	Group(1)	ReLU
64	64	5	2	0	Group(1)	ReLU
64	64	5	2	0	Group(1)	ReLU
64	16	1	1	0	None	None

In the last row, the first column refers to the manipulated mixture rendered by $\hat{s}_{\text{sum}}$, and the rest are the sources extracted from the mixture using the original queries (that is, the evaluation protocol for "Mixture Rendering" in Table 1). We can observe that the attributes are successfully manipulated. The second column combines the first source's timbre with the second's pitch, and the third column combines the second's timbre with the first's pitch. The third source is unchanged.

Furthermore, the left side of Fig. 5 shows that $\mathcal{L}_{\text{BT}}$ counteracts the over-regularisation effect caused by the standard Gaussian prior in the timbre latent space. Without the loss term, the sampling of the timbre latent is excessively noisy due to the large variance $\sigma_{\phi_\tau}(\cdot)$ of the Gaussian posterior.

C.2 The Latent Diffusion Model

Evaluation Given a reference mixture and its extracted set of pitch and timbre latents $\{\tau^{(i)}, \nu^{(i)}\}_{i=1}^{N_s}$, we arbitrarily replace the timbre latents $\{\tau^{(i)}\}_{i=1}^{N_s}$ with those from another target mixture. We expect that the four instruments of the reference mixture are replaced by those of the target mixture and that the target instruments play the original melodies of the reference instruments. We ensure that each instrument is swapped for the instrument labelled as the same SATB part in the target mixture, so that the target instruments play melodies that match their pitch range.

We sample $\{\hat{z}_s^{(i)}\}_{i=1}^{N_s}$ from $\mathcal{N}(0, \mathrm{I})$ conditioned on the manipulated set of timbre and pitch latents (Eq. (24)) in $T = 1000$ steps. We expect that feeding $\hat{x}_s^{(i)}$ to pre-trained instrument and pitch classifiers produces the targeted instrument and the original reference melody, respectively. Because the mixture is obtained by $\hat{x}_m = \sum_{i=1}^{N_s} \hat{x}_s^{(i)}$ in the audio domain, the evaluation of disentanglement suggests the results of mixture rendering. The results have been reported in Table 2.

Table 7 outlines the architecture of the instrument classifier used for the evaluation of disentanglement. The last layer is linearly projected to the logit of the 13 classes of the instruments. For the pitch classification, we employ Crepe, the state-of-the-art monophonic pitch extractor [24].[2] We use a public repository [3] to measure the FAD.

D Limitations

The main limitation of DisMix is the requirement of the queries that have to match the timbre characteristics or the instrument identity of constituent instruments of a mixture. This is a consequence of relying on the query-based source separation framework [27, 28, 52]. Few-shot learning has been shown to be a promising approach to relax the characteristics of the query example [52].

Based on the well defined instrument clusters in the left side of Fig. 2 and the minimisation of the Barlow Twin loss $\mathcal{L}_{\text{BT}}$, it is likely that we can use a post-hoc approach to find similarly discriminative clusters in the feature space of queries, e.g., using k-means clustering. We can then leverage the cluster means in place of queries during test time.

For future work, we could explore combining with DisMix a promising self-supervised method for source separation [41] to avoid the requirement of queries. We also consider unsupervised methods such as MusicSlots [15]. However, the model has only been evaluated by the simplistic dataset that we use for the simple case study.

[2]`https://github.com/maxrmorrison/torchcrepe`

[3]`https://github.com/gudgud96/frechet-audio-distance/tree/main`