FIGARO: CONTROLLABLE MUSIC GENERATION USING EXPERT AND LEARNED FEATURES

Dimitri von Rütte[*], **Luca Biggio**[*], **Yannic Kilcher, Thomas Hofmann**
Department of Computer Science, ETH Zürich
dimitri.vonrutte@inf.ethz.ch

ABSTRACT

Recent symbolic music generative models have achieved significant improvements in the quality of the generated samples. Nevertheless, it remains hard for users to control the output in such a way that it matches their expectation. To address this limitation, high-level, human-interpretable conditioning is essential. In this work, we release FIGARO, a Transformer-based conditional model trained to generate symbolic music based on a sequence of high-level control codes. To this end, we propose *description-to-sequence* learning, which consists of automatically extracting fine-grained, human-interpretable features (the *description*) and training a sequence-to-sequence model to reconstruct the original sequence given only the description as input. FIGARO achieves state-of-the-art performance in multi-track symbolic music generation both in terms of style transfer and sample quality. We show that performance can be further improved by combining human-interpretable with learned features. Our extensive experimental evaluation shows that FIGARO is able to generate samples that closely adhere to the content of the input descriptions, even when they deviate significantly from the training distribution.

1 INTRODUCTION

Music is a fascinating subject that surrounds us constantly, being a source of inspiration and canvas for imagination to many. To some, creating music is a topic worthy of dedicating one's life to, which is a testament to the artistry and mastery involved. While composition is an intricate form of art that requires a deep understanding of the human experience and domain knowledge, the idea of devising a systematic or algorithmic approach to music creation has been around for centuries (Nierhaus, 2009). With the advent of deep learning, automatic music generation has witnessed renewed interest (Hernandez-Olivan & Beltran, 2021). Especially the Transformer architecture (Vaswani et al., 2017), which has seen applications to many Machine Learning domains (Brown et al., 2020; Dosovitskiy et al., 2021; Lample & Charton, 2019; Biggio et al., 2021), has proven to be a powerful tool for musical sequence modelling. Initial breakthroughs by Huang et al. (2018) and Payne (2019) applied language modelling techniques to symbolic music to achieve state-of-the-art music generation. Though these models were capable of some form of conditional generation (e.g. melody or genre conditioning), other conditioning mechanisms and different types of control have since been proposed (Ens & Pasquier, 2020; Choi et al., 2020; Wu & Yang, 2021).

As deep generative models are improving and producing more and more realistic samples, it remains an area of active research how humans can interact with these models and steer them to generate a desirable result. Recent efforts in text-to-image generation (Ramesh et al., 2022; Saharia et al., 2022) have shown the potential in usability and artistic applications of human-interpretable controllable generative models. Whereas text-based conditioning has yielded human-interpretable control for image generation, the same conditioning mechanisms are not easily applicable to music generation. We aim to extend this kind of control to other domains, in this case to music generation. As scale has proven to be key for achieving capable models, we cannot rely on scarce annotated data and instead propose a self-supervised objective, which we call *description-to-sequence* learning. We take inspiration from recent text-to-image approaches, but instead of a natural language description of the target, we automatically extract a sequence of high-level features (the *description*). These can

[*]Equal contribution

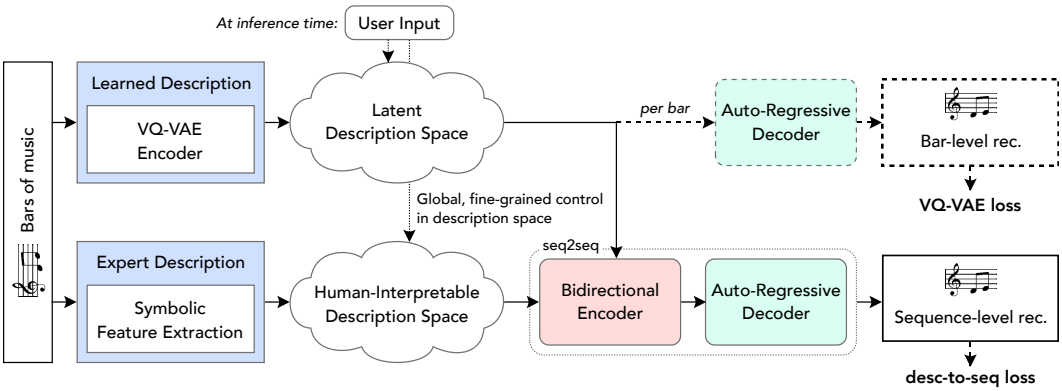


Figure 1: Overview of FIGARO. Dashed lines indicate components that are only used during training of the learned description.

either be hand-crafted using domain knowledge or learned. The description then serves as the input to a conditional model to reconstruct the original sequence. To this end, we define a *description function* which extracts said features from a given piece of music. The choice of description function determines the characteristics of the resulting model and serves as an inductive bias, allowing us to emphasize desirable properties such as human-interpretability or fine-grained control over instruments and chord progression. Note that the general nature of the proposed framework allows for applications to other domains despite our focus on symbolic music.

Our main contribution is FIGARO (FIne-grained music Generation via Attention-based, RObust control), a model trained on the proposed *description-to-sequence* objective by combining two separate description functions: 1) The hand-crafted *expert* description, which provides global context in the form of a high-level, human-interpretable sequence and 2) the *learned* description, where we use representation learning to extract salient features from the source sequence. The learned description is intended to amend the expert description with high-fidelity information in places where the latter might be incomplete, albeit at the cost of human-interpretability. The model is trained on conditional generation, mapping descriptions to music. An illustrated overview of the model is given in Figure 1. At inference time, users may interact with the model in human-interpretable description space. We provide a simple interface in the form of an online demo of our model.[1] We also release the source code and model weights for anyone to download and use freely.[2] Our secondary contribution is REMI+, an extension to the REMI input representation (Huang & Yang, 2020) which opens the way to multi-track, multi-time-signature music.

We evaluate FIGARO on its ability to adhere to the prescribed condition by comparing it to state-of-the-art methods for controllable symbolic music generation (Choi et al., 2020; Wu & Yang, 2021). We demonstrate empirically that our technique outperforms the state-of-the-art in controllable generation and sample quality. To evaluate sample quality, we employ subjective evaluation in the form of a listening study. We further demonstrate that FIGARO is robust with respect to distributional shifts in description space and performs well on constructed samples outside the training distribution, indicating that the proposed objective is effective at learning generalized concepts about the data.

2 Controllable Symbolic Music Generation

In the context of generative modelling, controllability is an important issue, as such models only become useful if the user is able to steer the generation process in a desired direction. This has recently been observed for text-to-image models and we intend to take a closer look at controllable music generation. We identify two different levels of controllability: global and fine-grained control. Global conditioning, where the generation is guided by a constant set of attributes that do not change during the generation process, is the most prevalent form of control. Examples of global control include prompt-based conditioning (Payne, 2019) or conditional decoding of latent representations

[1] Online demonstration of FIGARO is available on Google Colab (*https://tinyurl.com/28etxz27*). We recommend selecting a GPU environment for improved inference speed.

[2] Code and model weights are available through GitHub (*https://github.com/dvruette/figaro*).

Method	Input Rep.	Multi-Track	Multi-Sig.	Global Ctrl.	Fine-Grained Ctrl.
MIDI-VAE Brunner et al. (2018)	Pianoroll	✓	✓	✓	-
MuseNet (Payne, 2019)	MIDI-like	✓	✓	(✓)[3]	-
MMM (Ens & Pasquier, 2020)	MIDI-like	✓	✓	✓	(✓)[4]
Choi et al. (2020)	MIDI-like	-	✓	✓	(✓)[5]
MuseMorphose (Wu & Yang, 2021)	REMI	-	-	✓	✓
FIGARO (ours)	REMI+	✓	✓	✓	✓

Table 1: Generative capability and controllability comparison between different methods proposed in the literature. We compare our method to other state-of-the-art symbolic music generation methods on modelling capability (can the model generate multi-track/multi-time-signature music?) and controllability (can the generation be controlled on a global/fine-grained level?).

(Brunner et al., 2018). Fine-grained control is achieved when the generation process can be guided at any point in time, i.e. if the control attributes can be arbitrarily varied over time. Consider controlling what instruments are playing in the generated sequence as an example of global control in contrast to controlling what instruments are playing *at any point in time* as an example of fine-grained control. Note that fine-grained control also implies global control, as global control can be achieved by fixing the control attributes to some constant value. Fine-grained control is therefore a strictly more powerful property and seemingly harder to obtain, as is highlighted in Table 1.

2.1 Related Work

The capabilities of symbolic music generative models have been steadily improving with many recent contributions (Huang et al., 2018; Payne, 2019; Huang & Yang, 2020; Hsiao et al., 2021; Wu et al., 2022). This line of work focuses on improving the quality of generated samples but does not contribute substantially toward controllability. An exception to that is MuseNet (Payne, 2019), which allows some control through prompt-based conditioning with control tokens. Even still, prompt-based control is very limited, as control tokens are "forgotten" by the model once the generation advances beyond the initial context size.

Another line of work focuses on finding ways of controlling the generation process. VAE-based methods often achieve global control through latent conditioning vectors, enabling genre transfer (Brunner et al., 2018) capabilities or control over arousal (Tan & Herremans, 2020). Transformer auto-encoders have been used for melody conditioning (Huang et al., 2018; Choi et al., 2020) and encoding of musical style Choi et al. (2020). Ens & Pasquier (2020) present MMM which is capable of bar-level and track-level symbolic music inpainting. Finally, Wu & Yang (2021) introduce MuseMorphose for fine-grained attribute conditioning and latent space style editing. All of these approaches have some limitations that are highlighted in Table 1. Recent work tackling the problem of controllable generation is often either limited to single-track, single-time-signature music or to global control.

Fine-grained control has been a topic of interest in the recent literature (Choi et al., 2020; Wu & Yang, 2021; Di et al., 2021; Ferreira & Whitehead, 2021) and is an essential property when considering user-directed applications. In essence, fine-grained control is necessary to allow control over salient features in the generation, as saliency in music at least partly lies in how it changes over time. In addition, salient features may be impossible or prohibitively expensive to quantify directly (Choi et al., 2020; Ferreira & Whitehead, 2021), emphasizing the need for un- or self-supervised fine-grained control. Our proposed method provides both global and fine-grained control through description-to-sequence learning.

3 Description-to-Sequence Learning

Description-to-sequence learning is a self-supervised objective based on the reconstruction of a sequence given automatically extracted features of that same sequence (the *description*) as input. Conceptually, an information bottleneck is applied by mapping a sample to the description space and the model is trained to learn a probabilistic map from description space to sample space.

[3]While it is possible to control the style (via artist tags) and instruments of the generated sequence, this information is "forgotten" by the model due to context scrolling once the generation advances beyond the initial prompt.

[4]Fine-grained control is limited to changing the note density through bar-level inpainting.

[5]Fine-grained control is limited to optionally prescribing the melody of the generated sequence.

```
<bos>
  Bar_1 TimeSignature_4/4 NoteDensity_3 MeanPitch_14 MeanVelocity_19 MeanDuration_32
    Instrument_Drums Instrument_Piano Instrument_E-Piano Instrument_SlapBass
    Chord_E:maj Chord_F#:min7
  Bar_2 TimeSignature_4/4 NoteDensity_3 MeanPitch_15 MeanVelocity_18 MeanDuration_27
    Instrument_Drums Instrument_Piano Instrument_SlapBass Instrument_Guitar
    Chord_C#:min Chord_E:maj Chord_A:maj7
  Bar_3 ...
<eos>
```

Figure 2: Example of an expert description. The description contains information about time signature, note density, pitch, velocity and duration as well as which instruments and chords are played throughout each bar.

Description-to-Sequence Objective. Let V_{seq} denote a sequence vocabulary and V_{desc} a description vocabulary. Let $\mathbf{x} \in V_{\text{seq}}^*$ be an input sequence and $F : V_{\text{seq}}^* \to V_{\text{desc}}^*$ be a partition-wise feature extraction function that extracts descriptive features given a partition $x_1, \ldots, x_n$ of $\mathbf{x}$ such that $\mathbf{x} = \text{concat}(x_1, \ldots, x_n)$. We then define the description of $\mathbf{x}$ to be:

$$\mathbf{d} = \text{concat}(F(x_1), \ldots, F(x_n)) \tag{1}$$

For simplicity, we write $\mathbf{d} = F(\mathbf{x})$ in the following. The description-to-sequence objective is then given by the cross-entropy reconstruction loss of $\mathbf{x}$ given $F(\mathbf{x})$:

$$\mathcal{L}_{\text{rec}}(\phi) = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}}[-\log p_\phi(\mathbf{x}|F(\mathbf{x}))] \tag{2}$$

Here $\mathcal{X}$ is the true distribution of sequences and ϕ denotes the parameters of a sequence-to-sequence model. An illustrated overview of this objective is given in Figure 3.
Note that some level of fine-grained control is guaranteed by the way we partition $\mathbf{x}$ prior to feature extraction, as the length of partitions puts an upper bound on the receptive field of the description function.

3.1 Expert Description

We aim for a human-interpretable, fine-grained conditioning mechanism by using domain knowledge to automatically extract high-level features that are easy to understand for human experts and relevant from a compositional standpoint. We identify time signature, chords and instruments as such features and also consider note density, mean pitch, mean velocity and mean duration as auxiliary features quantifying musical style (Choi et al., 2020). All of these features are computed bar-per-bar and quantized according to a special-purpose vocabulary. Finally, all resulting tokens are concatenated into a sequence, which yields the expert description function $F_{\text{expert}}(\mathbf{x})$, of which an example is given in Figure 2. More details on the feature extraction algorithm and quantization can be found in Appendix C.

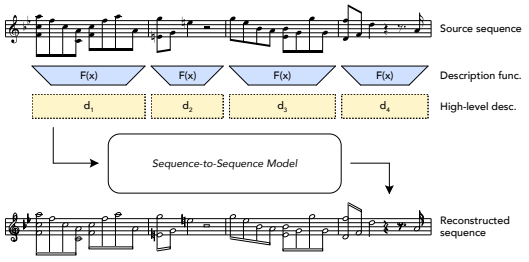


Figure 3: Schematic overview of description-to-sequence learning in the context of music.

3.2 Learned Description

While the expert description serves as a human-interpretable inductive bias, it is only able to capture fairly low-fidelity features by design and suffers from non-injectivity (i.e. there can be many sequences that map to the same description). In an attempt to alleviate some of these limitations, we use representation learning to extract features that maximize the information content about the underlying sequence. We choose the VQ-VAE framework as the basis for our learned description function as it produces discrete codes that allow for learning a prior sequence model and has been shown to be effective for various tasks such music generation in particular (Dhariwal et al., 2020). The latent representations given by this model is used as the output of the learned description function $F_{\text{learned}}(\mathbf{x})$. This way, we add high-fidelity information and reduce non-injectivity, albeit at the cost of human interpretability.
We later show that combining both descriptions yields better performance than using either one on its own (Section 6.2) and that the resulting model is fairly robust (Section 6.3).

4 EXPERIMENTAL SETUP

REMI+ Input Representation. The Revamped MIDI (REMI) representation (Huang & Yang, 2020) is a beat-based input representation, which has recently been shown to give better sample quality than traditional time-based representation. REMI represents notes with four consecutive tokens encoding note position, pitch, velocity and duration and also includes chord and tempo events.

In order to make it applicable to any general MIDI piece, we extend REMI by adding instrument and time-signature information, yielding REMI+. We add tokens encoding the time signature of a given bar at the beginning of each bar and instrument tokens at the beginning of each note, thus making it suitable for multi-time-signature, multi-track modelling tasks. We also increase the temporal resolution of note positions from 4 to 12 sub-beats per quarter note. This allows for precise quantization of triplets in addition to sixteenths, which was previously not possible. For exact details on quantization and tokenization, we refer to Appendix B. A limitation left unaddressed by REMI+ is the inability to encode expressive performances, which requires frequent shifts in tempo or small time offsets. Time-based representations are still preferred for that setting.

In our experiments, we tokenize MIDI files using the REMI+ input representation and train FIGARO on the proposed objective using the combination of the proposed *expert* and *latent* description functions.

Description-to-Sequence Model. We use the original Transformer auto-encoder (Vaswani et al., 2017) architecture with 4 encoder and 6 decoder layers as the sequence-to-sequence backbone. Three separate models are trained using the proposed description-to-sequence objective (Equation 2) using either $F_{\text{expert}}(\mathbf{x})$, $F_{\text{learned}}(\mathbf{x})$ or a combination of both descriptions. Combining descriptions is done by simply summing their respective learned token embeddings. The model architecture is kept deliberately simple, so as to be able to accurately assess the effectiveness of the various description functions. Further details and hyperparameters are given in Appendix A.

VQ-VAE Model. For our bar-level VQ-VAE model, we use the same Transformer auto-encoder backbone but with an additional quantization bottleneck consisting of 16 latent codes with a shared codebook of size 2048 between the encoder and decoder. The model is trained on reconstruction for individual bars. Note that training FIGARO is a two-stage process: In the first step, we train the VQ-VAE for feature extraction, which is later used to compute $F_{\text{learned}}(\mathbf{x})$. The VQ-VAE model is frozen during training of the description-to-sequence model.

Training Setup. We use the LakhMIDI dataset (Raffel, 2016) as training data in all of our experiments, which to the best of our knowledge is the largest publicly available symbolic music dataset. We use a 80%-10%-10% training-validation-test split. Each model is trained for 24 hours on 4 Nvidia GTX 2080 Ti GPUs. For evaluation, we generate samples conditioned on descriptions sampled from the test set, generating 32 bars for each sample. For training details, we refer to Appendix A.

5 PERFORMANCE EVALUATION

5.1 FLUENCY

We use perplexity (PPL) as a way to measure fluency and to compare the likelihood of different models in addition to task-specific metrics. The perplexity measures the likelihood of sequences while normalizing over the sequence length, which makes it better suited to comparing sequences of different lengths than the negative log-likelihood.

5.2 DESCRIPTION FIDELITY

We also quantitatively evaluate the fidelity of generated sequences to the given condition. Let $\mathbf{x}$ denote a test sample and $F(\mathbf{x})$ its description. Then we generate $\hat{\mathbf{x}}$ by sampling the model conditioned on $F(\mathbf{x})$ and examine $\mathbf{x}$ and $\hat{\mathbf{x}}$ for similarity. Metrics are computed as an empirical estimate over the test distribution. More details and exact formulas are given in Appendix D.

Accuracy. We compute accuracy metrics for categorical values, namely for instruments, chords and time signature. Instruments and chord are multi-label features for which we compute the mean F_1 score. As there is only one time signature per bar, we compute the standard accuracy score.

Macro Overlapping Area. Previous work has used the overlapping area (OA) metric to quantify similarity between two musical sequences for a given feature (Choi et al., 2020; Wu & Yang, 2021). However, we find that the standard OA metric fails to take the order of the sequences into account, as feature histograms are computed over the entire sequence. For example, a sequence has maximal overlapping area with itself in reverse order, even though the resulting sequences will sound completely different. To alleviate this limitation, we propose the macro overlapping area (MOA), which partition-wise computes the overlap in the distributions of a given feature, taking sequential order into account. We use the MOA metric to compute similarity in pitch, velocity and duration between ground truth and reconstruction. The exact definition of our MOA metric is given in Appendix D.

Normalized Root-Mean-Square Error. We compute the normalized RMSE (NRMSE) for bar-wise note density. This helps compare similarity accross different feature magnitudes.

Cosine Similarity. We also evaluate chroma and grooving similarity as a way to quantify similarity in sound and rhythm as proposed by Wu & Yang (2021). We compute bar-wise cosine similarity for the chroma vectors (Fujishima, 1999) and grooving vectors (Dixon et al., 2004).

6 EXPERIMENTS

Recent work on controllable generation has largely focused on single-track symbolic music and to the best of our knowledge, there are no competitive baselines for multi-track music generation with fine-grained control. To make due, we train two state-of-the-art methods for single-track controllable generation (Choi et al., 2020; Wu & Yang, 2021) on the REMI+ input representation, which allows us to apply them to a multi-track setting. We also train an unconditional baseline based on Music Transformer (Huang et al., 2018), which acts as a sanity check: It has no additional information about the target sequence and essentially performs "random guessing", i.e. it uniformly samples a sequence from the training distribution. If any conditional model performs worse than unconditional on similarity metrics, it is an indication for mode collapse (Dieng et al., 2018). Indeed, although the training of MuseMorphose (Wu & Yang, 2021) initially appeared to be successful, quantitative analysis revealed that the model had suffered mode collapse, as evidenced by the small chord and instrument entropy in Table 2 and the worse-than-unconditional performance on some of the style transfer metrics in Table 3a. As this model was originally designed for modelling single-track piano music and the LakhMIDI dataset constitutes a quite drastic domain shift, we instead focus on comparison to Choi et al. (2020). Details on baseline training are given in Appendix E.

6.1 HUMAN INPUT

The question of how well FIGARO can handle human-generated descriptions is both important and tricky to evaluate quantitatively, as both F_{expert} and especially F_{learned} require time and domain knowledge to create (albeit substantially less than composing a piece of music from scratch). We therefore demonstrate the generative capabilities of our model in an interactive demo[6] based on F_{expert}. We provide four hand-crafted examples, namely a simple four-bar piano progression, an 8-bar description containing uncommon instruments, a 9-bar description based on a well-known Jazz standard and a longer 20-bar description demonstrating control over multiple features in parallel with fine-grained instructions over instruments, chords, note density and mean pitch. These example are meant to show several attributes of FIGARO, namely its level of controllability and its ability to generate meaningful outputs even when input descriptions are human-made and present features that rarely appear in the training set. The reader is invited to generate samples based on these descriptions or to come up with their own creations. Demonstrating the use of F_{learned} on human input is less straight-forward as in the absence of a prior model, latent codes need to be extracted from an existing piece of music. We approximate this and the previous setting by using pieces of music from the test set to extract descriptions and perform quantitative evaluation on style transfer and controllable generation in Section 6.2 and 6.3 respectively.

6.2 STYLE TRANSFER

In this experiment, we extract conditioning sequences by feeding samples drawn from the test set uniformly at random through neural bottlenecks in the case of MuseMorphose, Choi et al. (2020) and

[6]Online demonstration of FIGARO on Google Colab (*https://tinyurl.com/28etxz27*)

F_{learned} or a symbolic bottleneck in the case of F_{expert}. New samples are then generated based on the condition and closeness in style to the original sequence is evaluated using the metrics from Section 5. Note that many of these metrics are directly present in F_{expert} and FIGARO (expert) expectedly does well in that regard. The fact that the results obtained with the expert description adhere to the content of the description itself is a highly desirable feature for controllable music generation. However, it also performs well on chroma and grooving similarity, both of which are not directly present. This shows that the expert description can act as a strong inductive bias and helps guide the generation process even beyond what information is directly represented. By using this description, the degree of control exerted by the user is enhanced and the results match expectations. We also observe that adding learned features further improves performance across the board, with FIGARO beating FIGARO (expert) in every category. The success of this hybrid approach means that we can preserve the interpretability and inductive bias, yet increase the fidelity of the generated music by exploiting black-box AI. Our expert description and hybrid models significantly outperform all baselines, and the learned description model outperforms Choi et al. (2020) on most metrics by a slight margin. The difference in performance is explained by the conditioning used for Choi et al. (2020), where the conditioning vector is temporally aggregated over the entire sample with any style progression throughout the sequence being lost. We provide a non-cherrypicked collection of samples and encourage the reader to get an impression of the quality and diversity of the music by listening to some of them on SoundCloud. The full list of results is provided in Table 3a.

Model	ΔH_{inst}	ΔH_{chord}
Ground truth	(3.763)	(4.077)
Unconditional	-0.521	-0.328
Choi et al. (2020)	-0.116	-0.128
Wu & Yang (2021)	-1.954	-1.657
FIGARO (expert)	-0.031	-0.028
FIGARO (learned)	0.062	0.008
FIGARO	-0.102	-0.006

Table 2: Difference in instrument entropy H_{inst} and chord entropy H_{chord} between ground truth and modelled distributions. Model entropies are empirical estimates over samples from the style transfer task. Ground truth values are absolute, deltas are relative to ground truth.

6.3 Controllable Generation

To evaluate the robustness of our model with respect to out-of-distribution conditions, we combine two sequences $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ (either splicing or mixing the descriptions) and generate samples conditioned on the resulting descriptions. This way, we create examples that have unusual transitions (in the case of splicing) or conflicting information (in the case of mixing) and are not part of the training distribution in general.

Description splicing. For this experiment, we take 16 bars of each sequence and concatenate them into a novel description $\tilde{\mathbf{x}} = b_1^{(1)} \parallel \cdots \parallel b_{16}^{(1)} \parallel b_{17}^{(2)} \parallel \cdots \parallel b_{32}^{(2)}$. We then input the description $F(\tilde{\mathbf{x}})$ to the model and sample the output distribution to generate a "medley" of the input sequences. Importantly, none of the models are finetuned for this specific task. We see a drop in performance in all evaluation metrics for every model compared to the style transfer task, which is expected due to distributional shifts in the data. However, FIGARO and FIGARO (expert) drop significantly less than FIGARO (learned) and Choi et al. (2020), showing that the expert description provides a strong inductive bias and is robust with respect to distribution shifts. We sometimes even observe FIGARO turning the hard cut-off points from bar-splicing into smooth transitions, which can be attributed to its capacity to pay attention to future bars and anticipate significant changes in style. The complete list of results is available in Table 3b. We omit Wu & Yang (2021) from these experiments as we do not expect competitive performance due to mode collapse.

Description mixing. Description mixing is done by combining $F_{\text{expert}}(\mathbf{x}^{(1)})$ with $F_{\text{learned}}(\mathbf{x}^{(2)})$ and thus is a scenario that is only applicable to FIGARO, the only model that takes both description types as input. Note that mixing descriptions in this way potentially provides conflicting information to the model as the two sequences might not agree with each other on certain features which might decrease sample quality or lead to a collapse of the model's performance in the worst case. Conflicting information also makes quantitative evaluation more difficult, which is why we focus on qualitative evaluation. We find that mixing descriptions in this way is not detrimental to the sample quality and that the model relies on different descriptions for different attributes of the generated sample. In general terms, the model seems to rely on the expert description for attributes including instruments, harmonics and time signature (one might call these "low-fidelity" attributes) and on the learned

Model	Fluency	Accuracy			Fidelity					
	PPL ↓	I ↑	C ↑	TS ↑	ND ↓	P ↑	V ↑	D ↑	s_c ↑	s_g ↑
Unconditional	1.988	.191	.048	.751	2.192	.563	.153	.312	.306	.510
Choi et al. (2020)	2.049	.658	.184	.908	1.679	.646	.574	.484	.514	.688
Wu & Yang (2021)	2.094	.179	.050	**1.00**	0.873	.492	.050	.207	.312	.529
FIGARO (expert)	1.913	**.957**	.561	**.996**	0.319	.759	.658	.514	.712	.637
FIGARO (learned)	1.973	.594	.195	.969	0.738	.701	.653	.546	.544	.697
FIGARO	**1.705**	**.960**	**.593**	**.997**	**0.238**	**.827**	**.735**	**.748**	**.790**	**.853**

(a) Perplexity and similarity metrics for style transfer on LakhMIDI. Best values are highlighted.

Model	Fluency	Accuracy			Fidelity					
	PPL ↓	I ↑	C ↑	TS ↑	ND ↓	P ↑	V ↑	D ↑	s_c ↑	s_g ↑
Choi et al. (2020)	2.213	.441	.129	.808	1.407	.603	.396	.448	.437	.643
FIGARO (expert)	1.824	**.944**	**.524**	**.992**	0.384	.741	.559	.497	.705	.575
FIGARO (learned)	2.186	.381	.128	.829	0.831	.649	.424	.478	.446	.614
FIGARO	**1.782**	.917	.514	**.988**	**0.335**	**.807**	**.702**	**.694**	**.748**	**.744**

(b) Description splicing perplexity and similarity metrics. Best values are highlighted.

Model	Fluency	Accuracy			Fidelity					
	PPL ↓	I ↑	C ↑	TS ↑	ND ↓	P ↑	V ↑	D ↑	s_c ↑	s_g ↑
FIGARO (expert)	**1.894**	**.955**	**.553**	**.996**	**0.360**	**.700**	**.646**	**.434**	**.710**	**.639**
- w/o instruments	1.980	*.373*	**.568**	**1.00**	0.424	.674	.586	**.436**	.687	**.625**
- w/o chords	*2.023*	.895	*.100*	**.995**	0.564	.672	.603	.413	*.294*	.615
- w/o meta info.	1.966	.908	.536	*.795*	*0.878*	*.574*	*.205*	*.334*	.636	*.584*

(c) Ablation study perplexity and similarity metrics. Best/worst values are highlighted/emphasized.

Table 3: We compare our models to the unconditional baseline based on Huang et al. (2018), (Choi et al., 2020) and MuseMorphose (Wu & Yang, 2021) on perplexity (PPL) and similarity metrics. Similarity metrics include instrument F_1-score (I), chord F_1-score (C) and time signature accuracy (TS) as well as note density NRMSE (ND), pitch MOA (P), velocity MOA (V), duration MOA (D), chroma similarity s_c and grooving similarity s_g.

description for attributes including texture and rhythmic intensity ("high-fidelity" attributes). This is not surprising as the expert description is designed to make this kind of low-fidelity information readily available and the learned description is intended to "fill in the gaps" of the expert description. We invite the reader to examine how the models combines two pieces of music into one by listening to some of the samples on SoundCloud. We also provide evaluation metrics with respect to the two source samples in Appendix F.

6.4 Ablation Study

To evaluate which parts of the expert description are essential, we group it into three components: instruments, chords and meta-information. Instruments and chords include all tokens with information about instruments and chords respectively while all other tokens (time signature, note density and mean pitch, velocity and duration) are classified as meta-tokens. We train separate models with one part of the description removed and compare the performance to FIGARO (expert), which receives the full expert description as input.

As one would expect, removing each component reduces the performance significantly in the respective metrics, indicating that each component carries useful information not entirely inferable through the remaining components. Interestingly, our experiments show that removing any component slightly decreases the over-all performance even in metrics that we would not necessarily expect to be affected. Removing instrument information, for example, increases the error for note density, pitch, velocity and duration, indicating that the instruments also carry mutual information about those features. This seems plausible considering the fact that different styles (or genres) of music usually identifies a set of instruments that is common for said style. Similar arguments can be made for the other two components. The full list of results is available in Table 3c.

6.5 Subjective Evaluation

Finally, we evaluate the subjective quality of generated samples through a listening study, comparing our best model to the baselines. In this experiment we are not interested in how closely the generated

samples follows the prescribed condition, but instead try to quantify the perceived quality of samples by recording pairwise preferences. To this end, we have conducted a survey where participants were asked to indicate their preference between 20 second excerpts of two samples chosen uniformly at random.[7] In total, we gathered 7569 comparisons by 691 unique participants, whose expertise ranged everything between non-musicians and professional musicians, although no prior musical knowledge was required. In two types of questions, participants had to choose 1) between a real sample and a generated sample or 2) between two generated samples. Generated samples are taken from the style transfer task. Question type 1) ranks the different methods on how good generated samples are compared to real, human-composed music. In this respect, FIGARO beats all other baseline with a win rate of 39.3% compared to the next best model by Choi et al. (2020), which has a win rate of 33.2%. As evidenced by the fact that the unconditional baseline (Huang et al., 2018) and Choi et al. (2020) are very close in terms of win rates, sample quality and reconstruction accuracy are not necessarily correlated, highlighting the need for a dedicated evaluation of sample quality. Win rates of the different models are displayed in Figure 4 with 90% confidence intervals obtained through normal approximation. While Choi et al. (2020) is able to adhere to the prescribed condition as shown in Section 6.2, the quality of generated samples is approximately on par with the unconditional model. FIGARO on the other hand is able to surpass the unconditional baseline in sample quality, while also providing controllable generation capabilities. Question type 2) is used to construct a pairwise ranking of models by applying the Wilcoxon signed-rank test on the study results. In this ranking, our model beats each of the baselines with a p-value of $< 10^{-7}$. The complete ranking and test results can be found in Appendix G.

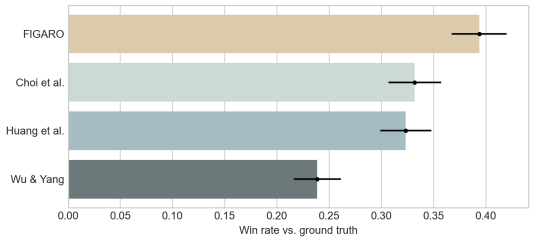


Figure 4: Win rates of generated samples against real samples. We compare FIGARO, Huang et al. (2018), Choi et al. (2020) and Wu & Yang (2021). Real samples are from the test set.

7 Discussion

Future Work. While our work represents a step towards high-quality symbolic music generation with human-interpretable control, we recognize three main avenues for future work. Firstly, the proposed model is strictly speaking not a generative model as creative input in the form of a description is still required. Training a prior model to generate such descriptions unconditionally could allow for faster generation with less human input required while still retaining the ability to influence the generation process in description space. Secondly, adding new description functions or extending the proposed ones could lead to further improved controllability and sample quality. Melody conditioning stands out as a useful attribute still lacking control in FIGARO. Lastly, applying the description-to-sequence objective to other domains such as natural language could potentially enable human-interpretable control in a way that is not currently possible. For example, the long-range conditioning ability could be interesting in the setting of story generation.

Conclusion. We present the self-supervised description-to-sequence objective and apply it in the context of symbolic music generation using expert and learned features. We train and release FIGARO, which combines two description functions in a model that achieves human-interpretable, fine-grained control on multi-track, multi-time-signature music while beating state-of-the-art controllable generation models both in terms of sample quality and style transfer. The proposed description functions complement each other: The expert description is a human-interpretable sequence that is easy to create and acts as a strong inductive bias to the model. The learned description is able to mitigate non-injectivity present in the expert description and amend it with high-fidelity, detailed information. In the case of conflicting descriptions, the model still produces plausible samples, thus preserving human-interpretability even in the presence of learned features. We can combine both descriptions to achieve state-of-the-art symbolic music generation with human-interpretable, fine-grained control. On a broader perspective, we hope that the proposed method is a step toward facilitating artists in their creative process as well as enabling amateurs to express themselves by lowering the barrier of entry to music creation and making the process faster and easier overall.

[7] Samples used in the study are available for download: *https://tinyurl.com/2ps5n2nv*

ETHICS STATEMENT

Automatic music generation may raise ethical concerns similar to those of large language models. The models may exhibit biases toward a certain style of music and may not represent the music of marginalized cultures accurately. In our case, the majority of training samples are western music, which is not necessarily representative of music at large. The model might also reproduce copyrighted material that is present in the training data and potentially generate samples that infringe on copyright law. Additionally, powerful music model capable of competing with human composers have the potential to create a big impact on the music industry. In this regard, we believe that our contribution is a step toward human-AI collaboration rather than competition.

REFERENCES

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 936–945. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/biggio21a.html.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer. In *arXiv:1809.07600 [cs, eess, stat]*, September 2018. URL http://arxiv.org/abs/1809.07600. arXiv: 1809.07600.

Kristy Choi, Curtis Hawthorne, Ian Simon, Monica Dinculescu, and Jesse Engel. Encoding Musical Style with Transformer Autoencoders. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 1899–1908. PMLR, November 2020. URL https://proceedings.mlr.press/v119/choi20b.html. ISSN: 2640-3498.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv: 1810.04805.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A Generative Model for Music. *arXiv:2005.00341 [cs, eess, stat]*, April 2020. URL http://arxiv.org/abs/2005.00341. arXiv: 2005.00341.

Shangzhe Di, Zeren Jiang, Si Liu, Zhaokai Wang, Leyan Zhu, Zexin He, Hongming Liu, and Shuicheng Yan. Video Background Music Generation with Controllable Music Transformer. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 2037–2045, October 2021. URL https://doi.org/10.1145/3474085.3475195.

Adji B. Dieng, Yoon Kim, Alexander M. Rush, and David M. Blei. Avoiding latent variable collapse with generative skip models, 2018. URL https://arxiv.org/abs/1807.04863.

Simon Dixon, Fabien Gouyon, and Gerhard Widmer. Towards Characterisation of Music via Rhythmic Patterns. *ISMIR*, 2004.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv:2010.11929 [cs]*, June 2021. URL http://arxiv.org/abs/2010.11929. arXiv: 2010.11929.

Jeff Ens and Philippe Pasquier. MMM : Exploring Conditional Multi-Track Music Generation with the Transformer. *arXiv:2008.06048 [cs]*, August 2020. URL http://arxiv.org/abs/2008.06048. arXiv: 2008.06048.

Lucas N. Ferreira and Jim Whitehead. Learning to Generate Music With Sentiment. *arXiv:2103.06125 [cs, eess]*, March 2021. URL http://arxiv.org/abs/2103.06125. arXiv: 2103.06125.

Takuya Fujishima. Real-time chord recognition of musical sound: A system using common lisp music. In *Proc. ICMC, Oct. 1999*, pp. 464–367, 1999.

Carlos Hernandez-Olivan and Jose R. Beltran. Music Composition with Deep Learning: A Review. *arXiv:2108.12290 [cs, eess]*, September 2021. URL http://arxiv.org/abs/2108.12290. arXiv: 2108.12290.

Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs. *arXiv:2101.02402 [cs, eess]*, January 2021. URL http://arxiv.org/abs/2101.02402. arXiv: 2101.02402.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music Transformer. *arXiv:1809.04281 [cs, eess, stat]*, December 2018. URL http://arxiv.org/abs/1809.04281. arXiv: 1809.04281.

Yu-Siang Huang and Yi-Hsuan Yang. Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions. *arXiv:2002.00212 [cs, eess, stat]*, August 2020. URL http://arxiv.org/abs/2002.00212. arXiv: 2002.00212.

Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. Improve Transformer Models with Better Relative Position Embeddings. *arXiv:2009.13658 [cs]*, September 2020. URL http://arxiv.org/abs/2009.13658. arXiv: 2009.13658.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. Fast Decoding in Sequence Models using Discrete Latent Variables. *arXiv:1803.03382 [cs]*, June 2018. URL http://arxiv.org/abs/1803.03382. arXiv: 1803.03382.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics, 2019.

Gerhard Nierhaus. Historical Development of Algorithmic Procedures. In *Algorithmic Composition: Paradigms of Automated Music Generation*, pp. 7–66. Springer, Vienna, 2009. ISBN 978-3-211-75540-2. doi: 10.1007/978-3-211-75540-2_2. URL https://doi.org/10.1007/978-3-211-75540-2_2.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning. *arXiv:1711.00937 [cs]*, May 2018. URL http://arxiv.org/abs/1711.00937. arXiv: 1711.00937.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv:1807.03748 [cs, stat]*, January 2019. URL http://arxiv.org/abs/1807.03748. arXiv: 1807.03748.

Christine Payne. MuseNet, April 2019. URL https://openai.com/blog/musenet/.

Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. URL https://arxiv.org/abs/2205.11487.

Hao Hao Tan and Dorien Herremans. Music fadernets: Controllable music generation based on high-level features via low-level feature modelling. 2020. doi: 10.48550/ARXIV.2007.15474. URL https://arxiv.org/abs/2007.15474.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008. Curran Associates, Inc., 2017.

Guowei Wu, Shipei Liu, and Xiaoya Fan. The power of fragmentation: A hierarchical transformer model for structural segmentation in symbolic music generation, 2022. URL https://arxiv.org/abs/2205.08579.

Shih-Lun Wu and Yi-Hsuan Yang. MuseMorphose: Full-Song and Fine-Grained Music Style Transfer with Just One Transformer VAE. *arXiv:2105.04090 [cs, eess]*, May 2021. URL http://arxiv.org/abs/2105.04090. arXiv: 2105.04090.

A Implementation Details

A.1 Description-to-Sequence Model

We largely follow the original paper for both of our Transformer auto-encoders. Unless specified differently, hyperparameters are left unchanged. One exception is the reduced context size of 256 tokens for the sake of training time. In the same spirit, we also reduce the number of encoder layers to 4 but leave the number of decoder layers at 6. We use relative positional embeddings (Huang et al., 2020) as this has been shown to be beneficial for symbolic music generation (Huang et al., 2018). In addition to the positional embeddings, we also add a learned bar embedding and a learned beat-position embedding. This puts a limit on the maximum number of bars (512 in our case). With a $4/4$ time signature at 120 BPM this equates to 17 minutes of music, which we deem an acceptable limitation. In the case of using both descriptions, we simply add their token embeddings before passing it to the encoder. In total, the description-to-sequence model has 44.6 M trainable parameters.

In the presence of the learned description F_{learned}, the discrete latent codes from the VQ-VAE feature extractor are embedded using the frozen codebook from the VQ-VAE instead of re-learning new embeddings. This is done to ensure stable training and reduce the number of trainable parameters.

A.2 VQ-VAE Model

For our bar-level VQ-VAE model, we use the same Transformer auto-encoder backbone as for the description-to-sequence model. The final layer of the encoder is pooled as proposed by Devlin et al. (2019) and then fed through a vector-quantization bottleneck, quantizing it to 16 latent codes from a codebook of size 2048. We use a modified version of the sliced vector quantization scheme proposed by Kaiser et al. (2018) as our discretization bottleneck. Specifically, we decompose the latent representation $\mathbf{z}$ into 16 slices $\mathbf{z}_1, \ldots, \mathbf{z}_{16}$ and discretize each of them to a shared codebook $\mathcal{C}$ with $|\mathcal{C}| = 2048$ using the k-means discretization technique from the original paper (Oord et al., 2018). The latent vector is provided to the decoder through cross-attention. We use a linear layer before and after vector-quantization to project between model space and latent space. The VQ-VAE model has 43.7 M trainable parameters in total.

The model is trained on bar-level reconstruction by minimizing the canonical β-VQ-VAE loss without the auxiliary codebook loss (Oord et al., 2019) given by

$$\mathcal{L}_{\text{VQ}-\text{VAE}}(\phi) = \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[-\log p_\phi(\mathbf{x}|z_q(\mathbf{x})) + \beta\|z(\mathbf{x}) - \text{sg}(z_q(\mathbf{x})\|\right]$$

where $\text{sg}(x)$ denotes the stop-gradient operator, ϕ are the model parameters and $\beta = 0.02$. The codebook is updated using the EMA update step as proposed in the original paper. We employ random restarts (Dhariwal et al., 2020) to ensure optimal codebook usage. Bars with more tokens than the context size are truncated to fit.

A.3 Training Details

We train each model for 100k steps with a batch size of 512 sequences. Models are optimized using the Adam optimizer (Kingma & Ba, 2017) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-6}$ and 0.01 weight decay. We use the inverse-square-root learning rate schedule with initial constant warmup at 10^{-4} given by $10^{-4}/\max(1, \sqrt{n/N})$ where $N = 4000$ is the number of warmup steps. For additional details and hyperparameters, we refer to the source code.

B REMI+ Input Representation

We extend the original REMI representation (Huang & Yang, 2020) to make it suitable for general multi-track, multi-signature symbolic music sequences. We make modifications that add time signature and instrument information, we determine a unique order of events and use quantization schemes that allow for accurate representation for a diverse set of music. An example of a REMI+ sequence is given in Figure 5.

Bar Tokens. To provide the model with additional context, we include the index of the current bar in each bar token. This, along with the bar embedding, should help the model retrieve the correct information from the description and help determine the end of a piece at generation time.

Time signature. We add a time signature token at the beginning of each bar, indicating the time signature of the bar which is to follow. We adapt the convention that time signature changes may only happen at the beginning of a bar, which is commonly true in written music.

Instruments. We add instrument information as an additional token before each note event, indicating which instrument will play the following note.

Order of events. In theory, the order of notes within each bar can be arbitrary without compromising the validity of the sequence. But to make the modelling task easier, we define a unique and deterministic order of events. Specifically, we sort events by (Bar, Position, EventType, Instrument, Pitch) in ascending order (valid event types are {Chord, Tempo, Note}). This order is unique since a given instrument can only ever play a single note with a given pitch at a given time[8].

Quantization. We largely follow Huang & Yang (2020) in quantization. The most significant deviation is the use of 12 note onset positions per quarter note instead of 4 as proposed in the original work. For example, there will be 48 unique note onset positions for the $4/4$ time signature and 36 note onset positions for the $3/4$ time signature. This allows both triplet and sixteenth notes to be quantized accurately, which is important when considering a diverse set of music.

Instruments and note pitches are not quantized as they are categorical variables with 128 possible values by the MIDI specification. Note velocity is quantized to 32 intervals in $[0, 128]$ and note duration is quantized to position intervals defined by the following mesh:

$$\begin{aligned}\mathcal{M} = &\{1, \dots, 12\} \cup \\ &\{12 + 3i \mid i \in (1, \dots, 4)\} \cup \\ &\{12 + 4i \mid i \in (1, \dots, 3)\} \cup \\ &\{24 + 6i \mid i \in (1, \dots, 4)\} \cup \\ &\{48 + 12i \mid i \in (1, \dots, 12)\} \cup \\ &\{192 + 24i \mid i \in (1, \dots, 24)\}\end{aligned}$$

This ensures single position accuracy up to quarter notes, then 16th and triplet accuracy up to half notes and 8th note accuracy up to a full note. To limit vocabulary size, we switch to quarter note steps up to 8 full notes and half note steps up to 16 full notes after that. Notes longer than 16 full notes are truncated to this length. Finally, tempo change events are discretized to 32 intervals in $[0, 240]$.

```
<bos>
Bar_1 TimeSignature_3/4
  Pos_0 Tempo_120
  Pos_0 Chord_C:min
  Pos_0 Instrument_Drums Pitch_36 Vel_90 Dur_0
  Pos_0 Instrument_Piano Pitch_64 Vel_85 Dur_4
  Pos_4 Instrument_Piano Pitch_66 Vel_85 Dur_4
Bar_2 TimeSignature_3/4
  Pos_0 Tempo_120
...
<eos>
```

Figure 5: Example sequence represented in the REMI+ representation. At the beginning of each bar time signature, tempo and the current chord are noted, after which each note is represented through five subsequent tokens.

C Expert Description Algorithm

Pseudocode for generating the expert description is given in Algorithm 1. We quantize note density, mean pitch and velocity to 32 linearly spaced intervals in $[0, 12]$, $[0, 128]$ and $[0, 128]$ respectively. Mean duration is quantized to 32 logarithmically spaced intervals in $[0, 128]$ positions (12 positions per quarter note). Chords are extracted with an adapted version of the Viterbi algorithm also used by Huang & Yang (2020).

[8]This is an assumption that could be violated in theory but does hold in practice.

Algorithm 1 ExpertDescription

input musical sequence $\mathbf{x}$
output description $\mathbf{d}$
$\mathbf{d} \leftarrow ()$
$b_1, \ldots, b_n \leftarrow$ PARTITIONINTOBARS($\mathbf{x}$)
for $b_i \in (b_1, \ldots, b_n)$ **do**
  $N \leftarrow \{n \mid n$ is a note with onset in $b_i\}$
  $I \leftarrow \{\texttt{inst} \mid \texttt{inst}$ is being played during $b_i\}$
  $C \leftarrow \{\texttt{chord} \mid \texttt{chord}$ is being played during $b_i\}$
  $q \leftarrow$ duration of b_i in quarter notes
  $\texttt{ts} \leftarrow$ time signature at beginning of b_i
  $\texttt{nd} \leftarrow \frac{|N|}{q}$
  $\texttt{mp} \leftarrow \frac{1}{|N|} \sum_{n \in N}$ PITCH(n)
  $\texttt{mv} \leftarrow \frac{1}{|N|} \sum_{n \in N}$ VELOCITY(n)
  $\texttt{md} \leftarrow \frac{1}{|N|} \sum_{n \in N}$ DURATION(n)
  Quantize nd, mp, mv and md
  $d_i \leftarrow (i, \texttt{ts}, \texttt{nd}, \texttt{mp}, \texttt{mv}, \texttt{md}) \parallel \text{list}(I) \parallel \text{list}(C)$
  $\mathbf{d} \leftarrow \mathbf{d} \parallel d_i$
end for
return $\mathbf{d}$

D EVALUATION METRICS

D.1 MACRO OVERLAPPING AREA

As used by previous work, the overlapping area (OA) metric does not consider the sequential order of the investigated feature, as feature histograms are computed over the entire sequence. For example, the overlapping area of $\mathbf{x}$ and $\text{reverse}(\mathbf{x})$ would be maximal, even though the reversed sequence does not sound like the original sequence in general.

To alleviate this limitation, we adapt the OA metric to also consider temporal order. Let $\mathbf{x}$ and $\mathbf{y}$ denote two musical sequences and let $b_i^{(\mathbf{x})}$ and $b_i^{(\mathbf{y})}$ denote the i-th bar of $\mathbf{x}$ and $\mathbf{y}$ respectively. We compute the overlap in feature distributions for each bar by fitting a Gaussian distribution to the feature under examination (e.g. note pitch) and compute the overlapping area between the two distributions. Let this overlap be given by $\text{overlap}(b_i^{(\mathbf{x})}, b_i^{(\mathbf{y})})$. Then the macro overlapping area (MOA) between $\mathbf{x}$ and $\mathbf{y}$ is given by

$$\text{MOA}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \text{overlap}\left(b_i^{(\mathbf{x})}, b_i^{(\mathbf{y})}\right)$$

D.2 NORMALIZED ROOT-MEAN-SQUARE ERROR

In order to normalize for different feature magnitudes between different samples, we compute the normalized RMSE (NRMSE) for bar-wise note density. Let x denote the ground truth, $\hat{x}$ denote the reconstruction and N denote the length of the sequences. Then the NRMSE is given by

$$\text{RMSE}(x, \hat{x}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{x}_i - x_i)^2}$$

$$\text{NRMSE}(x, \hat{x}) = \frac{\text{RMSE}(x, \hat{x})}{\text{mean}(x)}$$

D.3 COSINE SIMILARITY

Let $\mathbf{v}_i^{(\mathbf{x})}$ and $\mathbf{v}_i^{(\mathbf{y})}$ denote the chroma vector (Fujishima, 1999) or grooving vector (Dixon et al., 2004) for the i-th bar in $\mathbf{x}$ and $\mathbf{y}$ respectively. We then average the cosine similarity over the entire sequence

	Accuracy			Fidelity					
	I ↑	C ↑	TS ↑	ND ↓	P ↑	V ↑	D ↑	s_c ↑	s_g ↑
FIGARO	.960	.593	.997	.238	.827	.735	.748	.790	.853
Expert Similarity	**.819**	**.351**	**.993**	**.453**	.625	.354	.473	**.595**	.621
Learned Similarity	.288	.057	.753	.696	**.699**	**.670**	**.623**	.398	**.741**
Mean	.554	.204	.873	.575	.662	.512	.548	.497	.681
Best case	.819	.351	.993	.453	.699	.670	.623	.595	.741

Table 4: Description mixing performance on similarity metrics. "FIGARO" refers to the performance on unaltered descriptions. Expert/learned similarity refers to similarity of generated samples to the sequence from which the expert/learned description was extracted from. Highlights indicate which of the two descriptions was more closely adhered to in terms of the respective metric.

to get the chroma/grooving similarity:

$$\mathrm{sim}_{\mathbf{v}}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbf{v}_i^{(\mathbf{x})} \cdot \mathbf{v}_i^{(\mathbf{y})}}{\|\mathbf{v}_i^{(\mathbf{x})}\| \|\mathbf{v}_i^{(\mathbf{y})}\|}$$

E Benchmark Models

E.1 Huang et al. (2018)

We reimplement the model proposed by Huang et al. (2018) as an unconditional baseline. We train the model on the REMI+ representation to allow for direct comparison to our method. We use the improved relative attention from Huang et al. (2020) to eliminate any possible advantage arising from different attention mechanisms. We largely stick to the hyperparameters from our models, using 6 decoder layers with a hidden size of 512 and a filter size of 2048. Training and optimization hyperparameters are also the same as for our models.

E.2 Choi et al. (2020)

We reimplement the model proposed by Choi et al. (2020) and train it on the REMI+ representation to allow for direct comparison to our method. We again use the improved relative attention from Huang et al. (2020) and largely stick to the hyperparameters from the original paper, using 6 encoder and 6 decoder layers. Unlike the original work, we do not use data augmentation since the dataset is large enough and in order to allow for fair comparison between the models. Due to GPU memory constraints we reduce the context size from 2048 to 1024 and use an accumulated batch size of 16, ensuring stable training. Training and optimization hyperparameters are the same as for our models.

E.3 Wu & Yang (2021)

We train MuseMorphose on the REMI+ representation to allow for direct comparison to our method. Adapting the released implementation for our experiments, we reduce the context size from 1280 to 512 tokens due to GPU memory constraints but leave all other hyperparameters as they were proposed in the original paper. The model is trained until convergence (approx. 125k steps). We limit the training data to a subset of the entire dataset (20k samples) due to technical limitations. This is still considerably more training data than what was used in the original paper (1k samples) and should not affect performance significantly compared to using the full dataset.

F Description Mixing

Quantitative evaluation for this task is not straight-forward as there are two different "ground truth" samples with potentially conflicting features and it is unclear, which one the model should adhere to for which features. It is clear though that the model follows one of the descriptions more closely for some features, which points to some degree of separation of concerns between the two descriptions. Exact numbers are provided in Table 4.

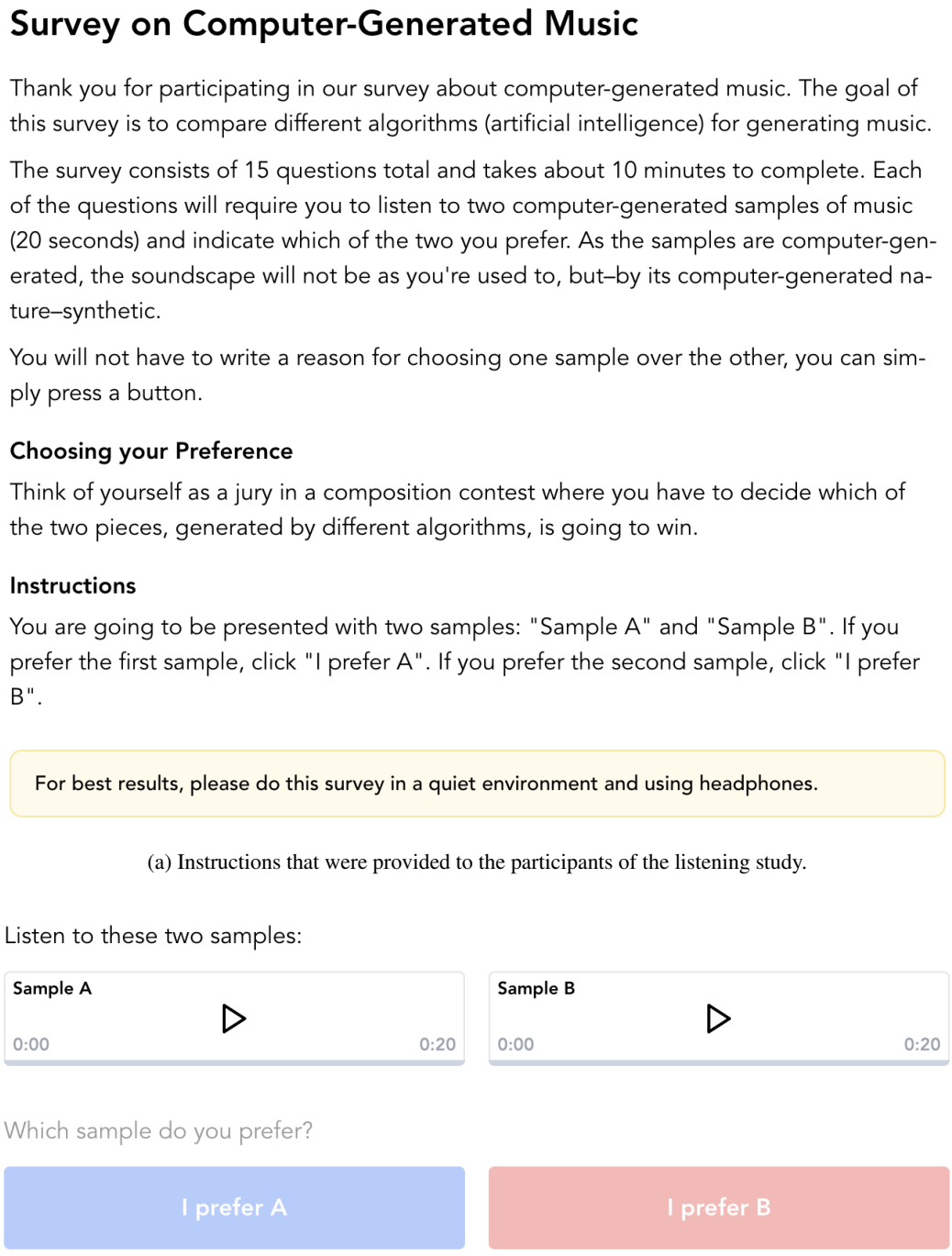
Model	Opponent	Winrate	p-value	N
1. Ground truth		0.669		2418
	FIGARO	0.605	$< 10^{-6}$	595
	Huang et al. (2018)	0.663	$< 10^{-15}$	605
	Choi et al. (2020)	0.647	$< 10^{-12}$	586
	Wu & Yang (2021)	0.756	$< 10^{-37}$	632
2. FIGARO		0.581		2439
	Huang et al. (2018)	0.609	$< 10^{-7}$	635
	Choi et al. (2020)	0.624	$< 10^{-9}$	631
	Wu & Yang (2021)	0.696	$< 10^{-20}$	578
3. Huang et al. (2018)		0.463		2467
	Choi et al. (2020)	0.543	0.0162	613
	Wu & Yang (2021)	0.581	$< 10^{-4}$	614
4. Choi et al. (2020)		0.447		2490
	Wu & Yang (2021)	0.589	$< 10^{-5}$	660
5. Wu & Yang (2021)		0.345		2484

Table 5: Ranking of the different methods by winrate according to the user study. The Wilcoxon signed-rank test is used to calculate pairwise ranking p-values. "Ground truth" denotes sampling the test set.

G Subjective Evaluation

The listening study includes 7569 comparisons by 691 participants, each averaging 11 answers. In each comparison, participants were presented with two different samples and were asked to indicate, which of the two they preferred. For the following ranking test, samples were chosen uniformly at random from two different generative models, the pair of which again was chosen at random. In this setup, we treat real samples as one possible model, for which we simply sample the test distribution. For the other models, we generate samples as described for the style transfer experiment (Section 6.2). Screenshots from the study are presented in Figure 6.

We apply the Wilcoxon signed-rank test to the results in order to establish a ranking of the different models. FIGARO beats each baseline except ground truth with a win rate of more than 60% and a p-value of less than 10^{-7}. Out of all methods, FIGARO also has the highest win rate against real samples with a win rate of 39.3%, which is a 6% advantage over the next best method. Rankings and corresponding p-values are reported in Table 5.



(a) Instructions that were provided to the participants of the listening study.

(b) Screenshot of the survey questions that were provided to the study participants after reading the instructions.

Figure 6: Screenshots of the listening study. Participants were first presented with the instructions (a) before answering 15 questions of the type presented in (b).