Encoding Musical Style with Transformer Autoencoders

Kristy Choi [1] [*] **Curtis Hawthorne** [2] **Ian Simon** [2] **Monica Dinculescu** [2] **Jesse Engel** [2]

Abstract

We consider the problem of learning high-level controls over the global structure of generated sequences, particularly in the context of symbolic music generation with complex language models. In this work, we present the Transformer autoencoder, which aggregates encodings of the input data across time to obtain a global representation of style from a given performance. We show it is possible to combine this global representation with other temporally distributed embeddings, enabling improved control over the separate aspects of performance style and melody. Empirically, we demonstrate the effectiveness of our method on various music generation tasks on the MAESTRO dataset and a YouTube dataset with 10,000+ hours of piano performances, where we achieve improvements in terms of log-likelihood and mean listening scores as compared to baselines.

1. Introduction

There has been significant progress in generative modeling, particularly with respect to creative applications such as art and music (Oord et al., 2016; Engel et al., 2017b; Ha & Eck, 2017; Huang et al., 2019a; Payne, 2019). As the number of generative applications increase, it becomes increasingly important to consider how users can interact with such systems, particularly when the generative model functions as a tool in their creative process (Engel et al., 2017a; Gillick et al., 2019) To this end, we consider how one can learn high-level controls over the global structure of a generated sample. We focus on the domain of symbolic music generation, where Music Transformer (Huang et al., 2019b) is the current state-of-the-art in generating high-quality samples that span over a minute in length.

The challenge in controllable sequence generation is

twofold. First, Transformers (Vaswani et al., 2017) and their variants excel as unconditional language models or in sequence-to-sequence tasks such as translation, but it is less clear as to how they can: (1) *learn* and (2) *incorporate* global conditioning information at inference time. This contrasts with traditional generative models for images such as the variational autoencoder (VAE) (Kingma & Welling, 2013) or generative adversarial network (GAN) (Goodfellow et al., 2014), both of which can incorporate global conditioning information (e.g. one-hot encodings of class labels) as part of their training procedure (Sohn et al., 2015; Sønderby et al., 2016; Isola et al., 2017; Van den Oord et al., 2016). Second, obtaining a "ground truth" annotation that captures all the salient features of a musical performance may be a prohibitively difficult or expensive task that requires domain expertise (Bertin-Mahieux et al., 2011). Thus even if conditioning was straightforward, the set of performance features that will be relevant for synthesizing a desired sample will remain ambiguous without descriptive tags to guide generation.

In this work, we introduce the Transformer autoencoder, where we aggregate encodings across time to obtain a holistic representation of the performance style in an unsupervised fashion. We demonstrate that this learned global representation can be incorporated with other forms of structural conditioning in two ways. First, we show that given a performance, our model can generate samples that are similar in style to the provided input. Then, we explore different methods to combine melody and performance representations to harmonize a new melody in the style of the given performance. We validate this notion of "perceptual similarity" through quantitative analyses based on note-based features of performances as well as qualitative user listening studies and interpolations. In both cases, we show that combining both global and fine-scale encodings of the musical performance allows us to gain better control of generation, separately manipulating both the style and melody of the resulting sample without the need for explicit labeling.

Empirically, we evaluate our model on two datasets: the publicly-available MAESTRO (Hawthorne et al., 2019) dataset, and a YouTube dataset of piano performances transcribed from 10,000+ hours of audio (Simon et al., 2019). We find that the Transformer autoencoder is able to generate not only performances that sound similar to the input, but
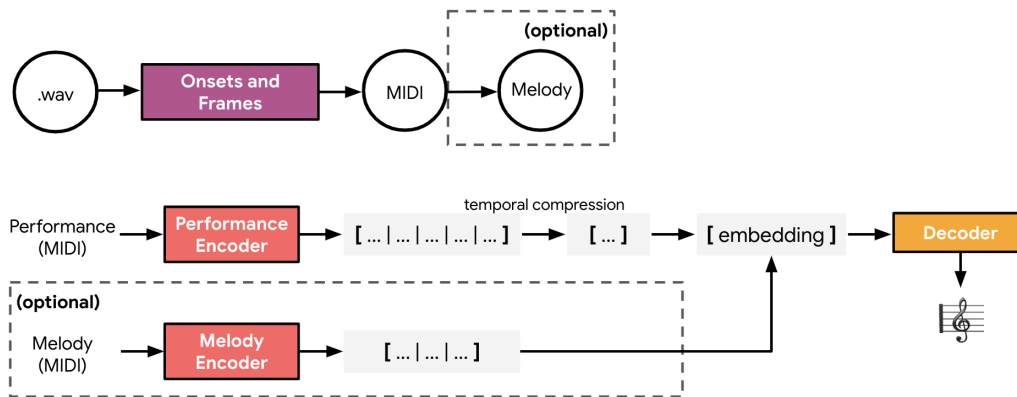
---
[1]Department of Computer Science, Stanford University *Work completed during an internship at Google Brain [2]Google Brain. Correspondence to: Kristy Choi <kechoi@cs.stanford.edu>.

Figure 1. A flowchart of the Transformer autoencoder. We first transcribe the `.wav` data files into MIDI using the Onsets and Frames framework, then encode them into performance representations to use as input. The output of the performance encoder is then aggregated across time and (optionally) combined with a melody embedding to produce a representation of the entire performance, which is then used by the Transformer decoder at inference time.

also accompaniments of melodies that follow a given style. In particular, we demonstrate that our model is capable of adapting to a particular musical style at test time *even in the case* where we have one single input performance.

2. Preliminaries

2.1. Data Representation for Music Generation

The MAESTRO (Hawthorne et al., 2019) dataset consists of over 1,100 classical piano performances, where each piece is represented as a MIDI file. The YouTube performance dataset is comprised of approximately 400K piano performances (over 10,000 hours) transcribed from audio (Simon et al., 2019). In both cases, we represent music as a sequence of discrete tokens, effectively formulating the generation task as a language modeling problem. The performances are encoded using the vocabulary as described in (Oore et al., 2018), which captures expressive dynamics and timing. This performance encoding vocabulary consists of 128 `note_on` events, 128 `note_off` events, 100 `time_shift` events representing time shifts in 10ms increments from 10ms to 1s, and 32 quantized `velocity` bins representing the velocity at which the 128 `note_on` events were played. We provide additional details of the data representation, encoding mechanism, and melody extraction procedure in the supplementary material.

2.2. Music Transformer

We build our Transformer autoencoder from Music Transformer, a state-of-the-art generative model that is capable of generating music with long-term coherence (Huang et al., 2019b). While the original Transformer uses self-attention to operate over absolute positional encodings of each token in a given sequence (Vaswani et al., 2017), Music

Transformer replaces this mechanism with *relative* attention (Shaw et al., 2018), which allows the model to keep better track of regularity based on event orderings and periodicity in the performance. (Huang et al., 2019b) propose a novel algorithm for implementing relative self-attention that is significantly more memory-efficient, enabling the model to generate musical sequences that span over a minute in length. For more details regarding the self-attention mechanism and Transformers, we refer the reader to (Vaswani et al., 2017; Parmar et al., 2018).

3. Generation with Transformer Autoencoder

3.1. Model Architecture

We leverage the standard encoder and decoder stacks of the Transformer as the foundational building block for our model, with minor modifications that we outline below.

Transformer Encoder: For both the performance and melody encoder networks, we use the Transformer's stack of 6 layers which are each comprised of a: (1) multi-head relative attention mechanism; and a (2) position-wise fully-connected feed-forward network. The performance encoder takes as input the event-based performance encoding of an input performance, while the melody encoder learns an encoding of the melody which has been extracted from the input performance. Depending on the music generation task (Section 3.2), the encoder output(s) are fed into the Transformer decoder. Figure 1 describes the way in which the encoder and decoder networks are composed together.

Transformer Decoder: The decoder shares the same structure as the encoder network, but with an additional multi-head attention layer over the encoder outputs. At each step of the generation process, the decoder takes in the output of the encoder, as well as each new token that was generated

in the previous timestep.

3.2. Conditioning Tasks and Mechanisms

Performance Conditioning and Bottleneck: We aim to generate samples that sound "similar" to a conditioning input performance. We incorporate a bottleneck in the output of the Transformer encoder in order to prevent the model from simply memorizing the input (Baldi, 2012). As shown in Figure 1, we mean-aggregate the performance embedding across the time dimension in order to learn a global representation of style. This mean-performance embedding is then fed into the autoregressive decoder, where the decoder attends to this global representation in order to predict the appropriate target. Although this bottleneck may be undesirable in sequence transduction tasks where the input and output sequences differ (e.g. translation), we find that it works well in our setting where we require the generated samples to be *similar in style* to the input sequence.

Melody & Performance Conditioning: Next, we synthesize any given melody in the style of a *different* performance. Although the setup is similar to the melody conditioning problem in (Huang et al., 2019b), we note that we also provide a conditioning performance signal, which makes the generation task more challenging. During training, we extract melodies from performances in the training set as outlined in (Waite, 2016), quantize the melody to a 100ms grid, and encode it as a sequence of tokens that uses a different vocabulary than the performance representation. For more details regarding the exact melody extraction procedure, we refer the reader to the supplement. We then use two distinct Transformer encoders (each with the same architecture) as in Section 3.1 to separately encode the melody and performance inputs. The melody and performance embeddings are combined to use as input to the decoder.

We explore various ways of combining the intermediate representations: (1) `sum`, where we add the performance and melody embeddings together; (2) `concatenate`, where we concatenate the two embeddings separated with a `stop` token; and (3) `tile`, where we tile the performance embedding across every dimension of time in the melody encoding. In all three cases, we work with the mean-aggregated representation of the input performance. We find that different approaches work better than others on some dataets, a point which we elaborate upon in Section 5.

3.3. Model Training

Input Perturbation: In order to encourage the encoded performance representations to generalize across various melodies, keys, and tempos, we draw inspiration from the denoising autoencoder (Vincent et al., 2008) as a means to regularize the model. For every target performance from which we extract the input melody, we provide the model with a *perturbed* version of the input performance as the conditioning signal. We allow this "noisy" performance to vary across two axes of variation: (1) `pitch`, where we artificially shift the overall pitch either down or up by 6 semitones; and (2) `time`, where we stretch the timing of the performance by at most $\pm 5\%$. Then for each new data point during training, a single noise injection procedure is randomly sampled from the cross product of all possible combinations of 12 pitch shift values and 4 time stretch values (evaluated in intervals of 2.5%). At test time, the data points are left unperturbed. In our experiments, we find that this augmentation procedure leads to samples that sound more pleasing (Oore et al., 2018).

Finally, the model is trained end-to-end with maximum likelihood: for a given sequence x of length n, we maximize $\log p_\theta(x) = \sum_{i=1}^{n} \log p_\theta(x_i | x_{<i})$ with respect to the model parameters θ. We emphasize that training is conducted in an autoencoder-like fashion. Specifically, for performance-only conditioning, the Transformer decoder is tasked with predicting the same performance provided to the encoder. For melody & performance conditioning, the Transformer autoencoder is trained to predict a new performance using the *combined* melody + performance embedding, where the loss is computed with respect to the input performance.

4. Performance Similarity Evaluation

As the development of a proper metric to quantify both the quality and similarity of musical performances remains an open question (Engel et al., 2019), we draw inspiration from (Yang & Lerch, 2018; Hung et al., 2019) to capture the style of a given performance based on eight features corresponding to its pitch and rhythm:

1. *Note Density (ND)*: The note density refers to the average number of notes per second in a performance: a higher note density often indicates a fast-moving piece, while a lower note density correlates with softer, slower pieces. This feature is a good indicator for rhythm.

2. *Pitch Range (PR)*: The pitch range denotes the difference between the highest and lowest semitones (MIDI pitches) in a given phrase.

3. *Mean Pitch (MP) / Variation of Pitch (VP)*: Similar in vein to the pitch range (PR), the average and overall variation of pitch in a musical performance captures whether the piece is played in a higher or lower octave.

4. *Mean Velocity (MV) / Variation of Velocity (VV)*: The velocity of each note indicates how hard a key is pressed in a musical performance, and serves as a heuristic for overall volume.

Model variation	MAESTRO	YouTube
Unconditional model with rel. attention (Huang et al., 2019b)	1.840	1.49
Performance autoencoder with rel. attention (ours)	**1.799**	**1.384**

Table 1. Note-wise test NLL on the MAESTRO and YouTube datasets. We exclude the performance autoencoder baseline (no aggregation) as it memorized the data (NLL = 0). Conditional models outperformed their unconditional counterparts.

Model variation	MAESTRO	YouTube
Melody-only Transformer with rel. attention (Huang et al., 2019b)	1.786	1.302
Melody & performance autoencoder with rel. attention, sum (ours)	**1.706**	1.275
Melody & performance autoencoder with rel. attention, concat (ours)	1.713	**1.237**
Melody & performance autoencoder with rel. attention, tile (ours)	1.709	1.248

Table 2. Note-wise test NLL on the MAESTRO and YouTube datasets with melody conditioning. We note that `sum` worked best for MAESTRO, while `concatenate` outperformed all other baselines for the YouTube dataset.

5. *Mean Duration (MD) / Variation of Duration (VD)*:
The duration describes for how long each note is pressed in a performance, representing articulation, dynamics, and phrasing.

4.1. Overlapping Area (OA) Metric

To best capture the salient features within the periodic structure of a musical performance, we used a sliding window of 2s to construct histograms of the desired feature within each window. We found that representing each performance with such relative measurements better preserved changing dynamics and stylistic motifs across the entire performance as opposed to a single scalar value (e.g. average note density across the entire performance).

Similar to (Yang & Lerch, 2018; Hung et al., 2019), we smoothed each feature's histogram by fitting a Gaussian distribution – this allowed us to learn a compact representation per feature through its mean μ and variance σ^2. Then to compare two performances, we computed the Overlapping Area (OA) between the Gaussian pdfs of each feature to quantify their similarity. The OA can be used to pinpoint feature-wise similarities between two performances, while the average OA across all features (OA$_{\text{avg}}$) can be used as a scalar-value summary to compare two performances together. We use both variants to quantify how similar two musical performances are in terms of their relative features.

Concretely, suppose we compare two performances A and B for the "pitch range" feature. If we model $A \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $B \sim \mathcal{N}(\mu_2, \sigma_2^2)$, and let c denote the point of intersection between the two pdfs (assuming without loss of generality that $\mu_1 > \mu_2$), the OA between A and B is:

$$OA(A, B) = 1 - \text{erf}\left(\frac{c - \mu_1}{\sqrt{2\sigma_1^2}}\right) + \text{erf}\left(\frac{c - \mu_2}{\sqrt{2\sigma_2^2}}\right) \quad (1)$$

where erf($\cdot$) denotes the error function erf(x) $= \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$. We found that other divergences such as the Kullback-Leibler (KL) divergence and the symmetrized KL were more sensitive to performance-specific features (rather than melody) than the OA. Empirically, we demonstrate that this metric identifies the relevant characteristics of interest in our generated performances in Section 5.

We note that for the melody & performance conditioning case, we performed similarity evaluations of our samples against the original performance from which the melody was extracted, as opposed to the melody itself. This is because the melody (a monophonic sequence) is represented using a different encoding and vocabulary than the performance (a polyphonic sequence). Specifically, we average two OA terms: (1) OA(source performance of extracted melody, generated sample) and (2) OA(conditioning performance, generated sample), as our final similarity metric. In this way, we account for the contributions of both the conditioning melody and performance sequence.

5. Experimental Results

Datasets: We used both the MAESTRO (Hawthorne et al., 2019) and YouTube datasets (Simon et al., 2019) for the experimental setup. We used the standard 80/10/10 train/validation/test split from MAESTRO v1.0.0, and augmented the dataset by 10x using pitch shifts of no more than a minor third and time stretches of at most 5%. We note that this augmentation is distinct from the noise-injection procedure referenced in Section 3: the data augmentation merely increases the size of the initial dataset, while the perturbation procedure operates *only* on the input performance signal to regularize the learned model. The YouTube dataset did not require any additional augmentation.

MAESTRO	ND	PR	MP	VP	MV	VV	MD	VD	Avg
Performance (ours)	**0.651**	**0.696**	**0.634**	**0.689**	**0.693**	**0.732**	**0.582**	**0.692**	**0.67**
Unconditional	0.370	0.466	0.435	0.485	0.401	0.606	0.385	0.529	0.46
YouTube Dataset									
Performance (ours)	**0.731**	**0.837**	**0.784**	**0.838**	**0.778**	**0.835**	**0.785**	**0.827**	**0.80**
Unconditional	0.466	0.561	0.556	0.578	0.405	0.590	0.521	0.624	0.54

Table 3. Average overlapping area (OA) similarity metrics comparing performance conditioned models with unconditional models. Unconditional and Melody-only baselines are from (Huang et al., 2019b). The metrics are described in detail in Section 4. The samples in this quantitative comparison are used for the listener study shown in the left graph of Figure 4.

MAESTRO	ND	PR	MP	VP	MV	VV	MD	VD	Avg
Melody & perf. (ours)	**0.650**	**0.696**	0.634	0.689	**0.692**	0.732	**0.582**	**0.692**	**0.67**
Perf-only (ours)	0.600	0.695	**0.657**	**0.721**	0.664	**0.740**	0.527	0.648	0.66
Melody-only	0.609	0.693	0.640	0.693	0.582	0.711	0.569	0.636	0.64
Unconditional	0.376	0.461	0.423	0.480	0.384	0.588	0.347	0.520	0.48
YouTube Dataset									
Melody & perf (ours)	**0.646**	**0.708**	0.610	0.717	**0.590**	**0.706**	**0.658**	**0.743**	**0.67**
Perf-only (ours)	0.624	0.646	0.624	0.638	0.422	0.595	0.601	0.702	0.61
Melody-only	0.575	0.707	**0.662**	**0.718**	0.583	0.702	0.634	0.707	0.66
Unconditional	0.476	0.580	0.541	0.594	0.400	0.585	0.522	0.623	0.54

Table 4. Average overlapping area (OA) similarity metrics comparing models with different conditioning. Unconditional and Melody-only baselines are from (Huang et al., 2019b). The metrics are described in detail in Section 4. The samples in this quantitative comparison are used for the listener study shown in the right graph of Figure 4.

Experimental Setup: We implemented the model in the Tensor2Tensor framework (Vaswani et al., 2017), and used the default hyperparameters for training: 0.2 learning rate with 8000 warmup steps, `rsqrt_decay`, 0.2 dropout, and early stopping for GPU training. For TPU training, we use AdaFactor with the `rsqrt_decay` and learning rate warmup steps to be 10K. We adopt many of the hyperparameter configurations from (Huang et al., 2019b), where we reduce the query and key hidden size to half the hidden size, use 8 hidden layers, use 384 hidden units, and set the maximum relative distance to consider to half the training sequence length for relative global attention. We set the maximum sequence length (length of event-based representations) to be 2048 tokens, and a filter size of 1024. We provide additional details on the model architectures and hyperparameter configurations in the supplement.

5.1. Log-Likelihood Evaluation

As expected, the Transformer autoencoder with the encoder bottleneck outperformed other baselines. In Tables 1 and 2, we see that all conditional model variants outperform their unconditional counterparts. For the melody & performance model, different methods of combining the embeddings work better for different datasets. For example,

`concatenate` led to the lowest NLL for the YouTube dataset, while `sum` outperformed all other variants for MAESTRO. We report NLL values for both datasets for the perturbed-input model variants in the supplement.

5.2. Similarity Evaluation

We use the OA metric from Section 4 to evaluate whether using a conditioning signal in both the (a) performance autoencoder (Perf-only) and (b) melody & performance autoencoder (Melody & perf) produces samples that are more similar in style to the conditioning inputs from the evaluation set relative to other baselines.

First, we sample 500 examples from the test set as conditioning signals to generate one sample per input. Then, we compare each conditioning signal to: (1) the generated sample and (2) an unconditional sample from the Music Transformer. We compute the similarity metric as in Section 4 pairwise and average over 500 examples. As shown in Tables 3 and 4, the performance autoencoder generates samples that have 48% higher similarity to the conditioning input as compared to the unconditional baseline for the YouTube dataset (45% higher similarity for MAESTRO).
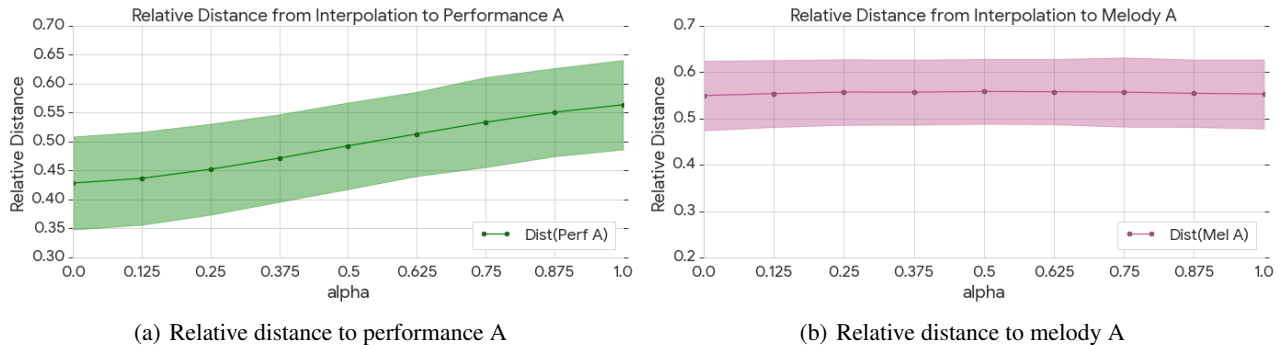
For the melody & performance autoencoder, we sample

(a) Relative distance to performance A

(b) Relative distance to melody A

Figure 2. For the YouTube dataset, relative distance from performance A ($\alpha = 1$) as α is slowly increased to 1.0 while the conditioned melody is fixed. As in (b), the relative distance to the conditioning melody with respect to a random performance remains fixed while the interpolation is conducted between performances A and B, suggesting that we can control for elements of style and melody separately.

717*2 distinct performances – we reserve one set of 717 for conditioning performance styles, and the other set of 717 we use to extract melodies in order to synthesize in the style of a *different* performance. We compare the melody & performance autoencoder to 3 different baselines: (1) one that is conditioned only on the melody (Melody-only); (2) conditioned only on performance (Perf-only); and (3) an unconditional language model. We find that the Melody & performance autoencoder performs the best overall across almost all features.

5.3. Latent Space Interpolations

Next, we analyze whether the Transformer autoencoder learns a semantically meaningful latent space through a variety of interpolation experiments on both model variants.

5.3.1. PERFORMANCE AUTOENCODER

We test whether the performance autoencoder can successfully interpolate between different input performances. First, we sample 1000 performances from the YouTube test set (100 for MAESTRO, due to its smaller size), and split this dataset in half. The first half we reserve for the original starting performance, which we call "performance A", and the other half we reserve for the end performance, denoted as "performance B." Then we use the performance encoder to embed performance A into z_A, and do the same for performance B to obtain z_B. For a range $\alpha \in [0, 0.125, \dots, 0.875, 1.0]$, we sample a new performance perf$_{new}$ that results from decoding $\alpha \cdot z_A + (1-\alpha) \cdot z_B$. We observe how the OA$_{avg}$ (averaged across all features) defined in Section 4 changes between this newly interpolated performance perf$_{new}$ and performances $\{A, B\}$.

Specifically, we compute the similarity metric between each input performance A and interpolated sample perf$_{new}$ for all 500 samples, and compute the same pairwise similar-

ity for each performance B. We then compute the normalized distance between each interpolated sample and the corresponding performance A or B, which we denote as: `rel_distance(perf A)` $= 1 - \frac{OA\_A}{OA\_A + OA\_B}$, where the OA is averaged across all features. We average this distance across all elements in the set and find in Figure 3 that the relative distance between performance A slowly increases as we increase α from 0 to 1, as expected. We note that it is not possible to conduct this interpolation study with non-aggregated baselines, as we cannot interpolate across variable-length embeddings. We find that a similar trend holds for MAESTRO as in Figure 2(a).
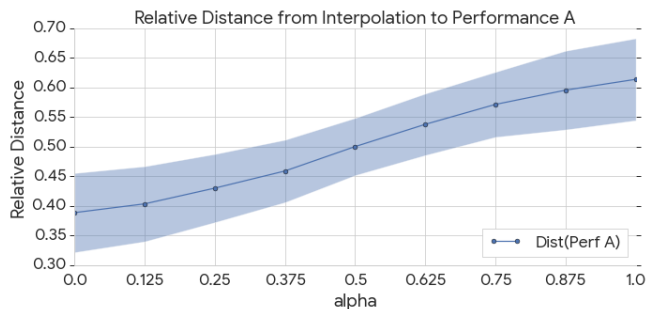

Figure 3. For the YouTube dataset, the relative distance from performance A ($\alpha = 1$) to the interpolated sample increases as α is slowly increased to 1.0.

5.3.2. MELODY & PERFORMANCE AUTOENCODER

We conduct a similar study as above with the melody & performance autoencoder. We hold out 716 unique melody-performance pairs (melody is not derived from the same performance) from the YouTube evaluation dataset and 50 examples from MAESTRO. We then interpolate across the different performances, while keeping the conditioning melody input the same across the interpolations.
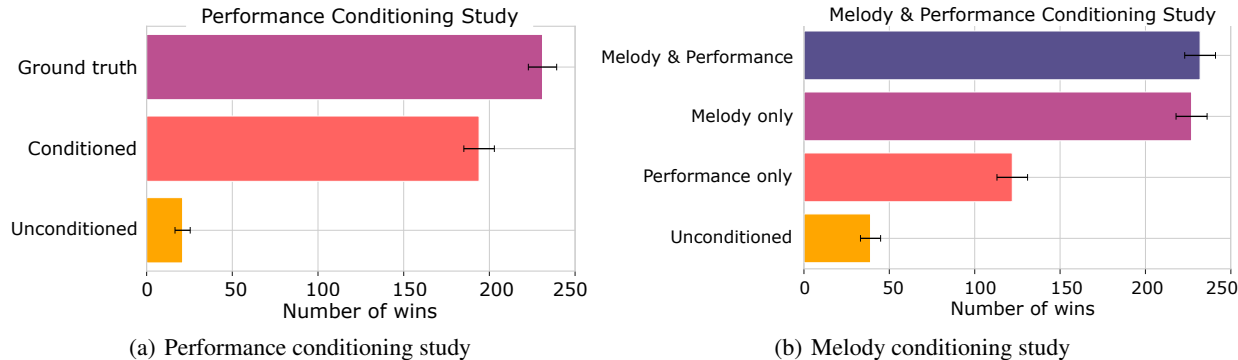
(a) Performance conditioning study (b) Melody conditioning study

Figure 4. Results of our listening studies, showing the number of times each source won in a pairwise comparison. Black error bars indicate estimated standard deviation of means.

As shown in Figure 2(a), we find that a similar trend holds as in the performance autoencoder: the newly-interpolated samples show that the relative distance between performance A increases as we increase the corresponding value of α. We note that the interpolation effect is slightly lower than that of the previous section, particularly because the interpolated sample is also dependent on the melody that it is conditioned on. Interestingly, in Figure 2(b), we note that the relative distance between the input performance from which we derived the original melody remains fairly constant across the interpolation procedure. This suggests that we are able to factorize out the two sources of variation and that varying the axis of the input performance keeps the variation in melody constant.

5.4. Listening Tests

To further evaluate the perceived effect of performance and melody conditioning on the generated output, we also conducted qualitative listening tests. Using models trained on the YouTube dataset, we conducted two studies for separate music generation tasks: one for the performance conditioning, and one for melody and performance conditioning.

5.4.1. PERFORMANCE CONDITIONING

For performance conditioning, we presented participants with a 20s performance clip from the YouTube evaluation dataset that we used as a conditioning signal. We then asked them to listen to two additional 20s performance clips and to use a Likert scale to rate which one sounded most similar in style to the conditioning signal. The sources rated by the participants included "Ground Truth" (a different snippet of the same sample used for the conditioning signal), "Conditioned" (output of the Performance Autoencoder), and "Unconditioned" (output of unconditional Music Transformer). We collected a total of 492 ratings, with each source involved in 328 distinct pair-wise comparisons.

5.4.2. MELODY AND PERFORMANCE CONDITIONING

For melody and performance conditioning, we similarly presented participants with a 20s performance clip from the YouTube evaluation dataset and a 20s melody from a different piece in the evaluation dataset that we used as our conditioning signals. We then asked each participant to listen to two additional 20s performance clips and to use a Likert scale to rate which sounded most like the conditioning melody played in the style of the conditioning performance. The sources rated by the participants included "Melody & Performance" (output of the Melody-Performance Autoencoder), "Melody only" (output of a model conditioned only on the melody signal), "Performance only" (output of a model conditioned only on the performance signal), and "Unconditioned" (output of an unconditional model). For this study, we collected a total of 714 ratings, with each source involved in 357 distinct pair-wise comparisons.

Figure 4 shows the number of comparisons in which each source was selected as being most similar in style to the conditioning signal. A Kruskal-Wallis H test of the ratings showed that there is at least one statistically significant difference between the models: $\chi^2(2) = 332.09$, $p < 0.05$ ($7.72e-73$) for melody conditioning and $\chi^2(2) = 277.74$, $p < 0.05$ ($6.53e-60$) for melody and performance conditioning. A post-hoc analysis using the Wilcoxon signed-rank test with Bonferroni correction showed that there were statistically significant differences between all pairs of the performance study with $p < 0.05/3$ and all pairs of the performance and melody study with $p < 0.05/6$ except between the "Melody only" and "Melody & Performance" models ($p = 0.0894$).

These results demonstrate that the performance conditioning signal has a clear, robust effect on the generated output: in the 164 comparisons between "Ground Truth" and "Conditioned", participants responded that they had a preference for "Conditioned" sample 58 times.

Although the results between "Melody-only" and "Melody & Performance" are close, this study demonstrates that conditioning with both melody and performance outperforms conditioning on performance alone, and they are competitive with melody-only conditioning, despite the model having to deal with the complexity of incorporating both conditioning signals. In fact, we find quantitative evidence that human evaluation is more sensitive to perceptual melodic similarity, as the "Performance-only" model performs worst – a slight contrast to the results from the OA metric in Section 5.2.

Our qualitative findings from the audio examples and interpolations, coupled with the quantitative results from the OA similarity metric and the listening test which capture different aspects of the synthesized performance, support the finding that the Melody & Performance autoencoder offers significant control over the generated samples. We provide several audio examples demonstrating the effectiveness of these conditioning signals in the online supplement at `https://goo.gl/magenta/music-transformer-autoencoder-examples`.

6. Related Work

Measuring music similarity: We note that quantifying music similarity is a difficult problem. We incorporate and extend upon the rich line of work for measuring music similarity in symbolic music (Ghias et al., 1995; Berenzweig et al., 2004; Slaney et al., 2008; Hung et al., 2019; Yang & Lerch, 2018) for our setting, in which we evaluate similarities between polyphonic piano performances as opposed to monophonic melodies.

Sequential autoencoders: Building on the wealth of autoencoding literature (Hinton & Salakhutdinov, 2006; Salakhutdinov & Hinton, 2009; Vincent et al., 2010), our work bridges the gap between the traditional sequence-to-sequence framework (Sutskever et al., 2014), their recent advances with various attention mechanisms (Vaswani et al., 2017; Shaw et al., 2018; Huang et al., 2019b), and sequential autoencoders. Though (Wang & Wan, 2019) propose a Transformer-based conditional VAE for story generation, the self-attention mechanism is shared between the encoder and decoder. Most similar to our work is that of (Kaiser & Bengio, 2018), which uses a Transformer decoder and a discrete autoencoding function to map an input sequence into a discretized, compressed representation. We note that this approach is complementary to ours, where a similar idea of discretization may be applied to the output of our Transformer encoder. The MusicVAE (Roberts et al., 2018) is a sequential VAE with a hierarchical recurrent decoder, which learns an interpretable latent code for musical sequences that can be used during generation time. This work builds upon (Bowman et al., 2015) that uses recurrence and an autoregressive decoder for text generation. Our Transformer autoencoder can be seen as a deterministic variant of the MusicVAE, with a complex self-attention mechanism based on relative positioning in both the encoder and decoder architectures to capture more expressive features of the data at both the local and global scale.

Controllable generations using representation learning: There is also considerable work on controllable generations, where we focus on the music domain. (Engel et al., 2017a) proposes to constrain the latent space of unconditional generative models to sample with respect to some predefined attributes, whereas we explicitly define our conditioning signal in the data space and learn a global representation of its style during training. The Universal Music Translation network aims to translate music across various styles, but is not directly comparable to our approach as they work with raw audio waveforms (Mor et al., 2018). Both (Meade et al., 2019) and MuseNet (Payne, 2019) generate music based on user preferences, but adopt a slightly different approach: the models are specifically trained with labeled tokens (e.g., composer and instrumentation) as conditioning input, while our Transformer autoencoder's global style representation is learned in an unsupervised way. We emphasize Transformer autoencoder's advantage of learning unsupervised representations of style, as obtaining ground truth annotations for music data may be prohibitively challenging.

7. Conclusion

We proposed the Transformer autoencoder for conditional music generation, a sequential autoencoder model which utilizes an autoregressive Transformer encoder and decoder for improved modeling of musical sequences with long-term structure. We show that this model allows users to easily adapt the outputs of their generative model using even a single input performance. Through experiments on the MAESTRO and YouTube datasets, we demonstrate both quantitatively and qualitatively that our model generates samples that sound similar in style to a variety of conditioning signals relative to baselines. For future work, it would be interesting to explore other training procedures such as variational techniques or few-shot learning approaches (Finn et al., 2017; Reed et al., 2017) to account for situations in which the input signals are from slightly different data distributions than the training set. We provide open-sourced implementations in Tensorflow (Abadi et al., 2016) at `https://goo.gl/magenta/music-transformer-autoencoder-code`.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

Baldi, P. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49, 2012.

Berenzweig, A., Logan, B., Ellis, D. P., and Whitman, B. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2): 63–76, 2004.

Bertin-Mahieux, T., Eck, D., and Mandel, M. Automatic tagging of audio: The state-of-the-art. In *Machine audition: Principles, algorithms and systems*, pp. 334–352. IGI Global, 2011.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Engel, J., Hoffman, M., and Roberts, A. Latent constraints: Learning to generate conditionally from unconditional generative models. *arXiv preprint arXiv:1711.05772*, 2017a.

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., and Simonyan, K. Neural audio synthesis of musical notes with wavenet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1068–1077. JMLR. org, 2017b.

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., and Roberts, A. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Ghias, A., Logan, J., Chamberlin, D., and Smith, B. C. Query by humming: musical information retrieval in an audio database. In *Proceedings of the third ACM international conference on Multimedia*, pp. 231–236, 1995.

Gillick, J., Roberts, A., Engel, J., Eck, D., and Bamman, D. Learning to groove with inverse sequence transformations. *arXiv preprint arXiv:1905.06118*, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Ha, D. and Eck, D. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1lYRjC9F7.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science*, 313 (5786):504–507, 2006.

Huang, C.-Z. A., Cooijmans, T., Roberts, A., Courville, A., and Eck, D. Counterpoint by convolution. *arXiv preprint arXiv:1903.07227*, 2019a.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music transformer: Generating music with long-term structure. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=rJe4ShAcF7.

Hung, H.-T., Wang, C.-Y., Yang, Y.-H., and Wang, H.-M. Improving automatic jazz melody generation by transfer learning techniques. *arXiv preprint arXiv:1908.09484*, 2019.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Kaiser, Ł. and Bengio, S. Discrete autoencoders for sequence models. *arXiv preprint arXiv:1801.09797*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Meade, N., Barreyre, N., Lowe, S. C., and Oore, S. Exploring conditioning for generative music systems with human-interpretable controls. *arXiv preprint arXiv:1907.04352*, 2019.

Mor, N., Wolf, L., Polyak, A., and Taigman, Y. A universal music translation network. *arXiv preprint arXiv:1805.07848*, 2018.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A.,

and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Oore, S., Simon, I., Dieleman, S., Eck, D., and Simonyan, K. This time with feeling: learning expressive musical performance. *Neural Computing and Applications*, pp. 1–13, 2018.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Shazeer, N., Ku, A., and Tran, D. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

Payne, C. Musenet, 2019. URL https://openai.com/blog/musenet/.

Reed, S., Chen, Y., Paine, T., Oord, A. v. d., Eslami, S., Rezende, D., Vinyals, O., and de Freitas, N. Few-shot autoregressive density estimation: Towards learning to learn distributions. *arXiv preprint arXiv:1710.10304*, 2017.

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., and Eck, D. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

Salakhutdinov, R. and Hinton, G. Deep boltzmann machines. In *Artificial intelligence and statistics*, pp. 448–455, 2009.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Simon, I., Huang, C.-Z. A., Engel, J., Hawthorne, C., and Dinculescu, M. Generating piano music with transformer. 2019. URL https://magenta.tensorflow.org/piano-transformer.

Slaney, M., Weinberger, K., and White, W. Learning a metric for music similarity. 2008.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. In *Advances in neural information processing systems*, pp. 3738–3746, 2016.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pp. 4790–4798, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.

Waite, E. Generating long-term structure in songs and stories, 2016. URL https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn.

Wang, T. and Wan, X. T-cvae: Transformer-based conditioned variational autoencoder for story completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 5233–5239. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/727. URL https://doi.org/10.24963/ijcai.2019/727.

Yang, L.-C. and Lerch, A. On the evaluation of generative models in music. *Neural Computing and Applications*, pp. 1–12, 2018.