SCOREPERFORMER: EXPRESSIVE PIANO PERFORMANCE RENDERING WITH FINE-GRAINED CONTROL

Ilya Borovik
Skoltech, Russia
ilya.borovik@skoltech.ru

Vladimir Viro
Peachnote GmbH, Germany
vladimir@peachnote.de

ABSTRACT

We present ScorePerformer, an encoder-decoder transformer with hierarchical style encoding heads for controllable rendering of expressive piano music performances. We design a tokenized representation of symbolic score and performance music, the Score Performance Music tuple (SPMuple), and validate a novel way to encode the local performance tempo in a local note time window. Along with the encoding, we extend a transformer encoder with multi-level maximum mean discrepancy variational autoencoder style modeling heads that learn performance style at the global, bar, beat, and onset levels for fine-grained performance control. To offer an interpretation of the learned latent spaces, we introduce performance direction marking classifiers that associate vectors in the latent space with direction markings to guide performance rendering through the model. Evaluation results show the importance of the architectural design choices and demonstrate that ScorePerformer produces diverse and coherent piano performances that follow the control input.

1. INTRODUCTION

Musical expression is the human touch that transforms a written piece of music into an emotionally moving experience. In musical interpretation and performance, the musician interprets a musical score and translates the intended expression through the control of the musical instrument, the sound of which conveys affect and emotion to the listener [1, 2]. However, effective control of musical instruments often requires considerable expertise and training, making musical expression less accessible than it could be.

Deep learning music performance models reduce the need for musical expertise and open up new ways to create and perform music [3, 4]. To render expressive performances of written music [5, 6], the models mix recurrent neural networks to learn temporal dependencies in music with variational autoencoders to encode performance style and enable controllable generation [7–11]. The models are trained on real and categorical score and performance features for aligned score and performance notes.

The related task of symbolic music generation is approached differently. Transformer models [12] are primarily utilized due to their ability to effectively learn long-term dependencies in music sequences [13–17]. The symbolic music is encoded as sequences of musical tokens, either individual [15, 18, 19] or stacked into tuples [16, 20]. Similar approaches could be applied to the task of rendering expressive music performances for written compositions.

Aiming to advance the research and make musical expression more accessible, we develop ScorePerformer [1], a piano music performance rendering model with interactive fine-grained performance style control. The model combines encoder and decoder transformers [12] with hierarchical maximum mean discrepancy variational autoencoders [21, 22] that encode performance style representations at the global, bar, beat, and onset levels.

To interpret the learned style embedding spaces, we train embedding classifiers that associate local performance contexts with written musical score direction markings. For each marking, we use the classifier predictions to compute the average delta vectors in the style space from negatively to positively classified style embeddings. These vectors provide quantified model control inputs to move the performance rendering per given direction marking.

For data encoding, we design a tokenized representation of score and performance music, a Score-Performance Music tuple (SPMuple). It introduces a local window onset tempo function that produces smoother and more robust tempos than inset-bar, -beat, or -onset tempo functions.

The experiments and evaluation results show that the model trained on the designed encoding successfully captures different performance styles, can sample diverse and coherent piano performances, and can be used for expressive performance rendering with fine-grained style control.

Our main contributions are:

1. We extend transformers for expressive piano performance rendering with hierarchical style encoding and control at the global, bar, beat, and onset levels;
2. We design a tokenized encoding for aligned score and performance music that proposes an efficient local tempo computation function;
3. We introduce performance direction classifiers to provide musical language-driven performance control by modifying the learned style latent spaces.

[1] Source code and demo are available at: https://github.com/ilya16/scoreperformer

2. RELATED WORK

Expressive Music Performance: Recent expressive music performance rendering models mainly utilize deep learning methods [3, 6]. Jeong et al. [8] and Maezawa et al. [9] use conditional variational autoencoders for performance style encoding and recurrent neural networks for expressive performance rendering. Rhyu et al. [11] allow performance style to be intuitively "sketched" by a set of learned latent representations. We propose to use transformers with self-attention mechanisms [12] to infer patterns in music performance and model its style through hierarchical style encoding heads.

Symbolic Music Generation: Symbolic music generation with deep learning [4] is dominated by transformers for learning long-term sequential musical patterns [13, 15–17, 23] and variational autoencoders for unsupervised style encoding and control [19, 23–26]. The models offer unconditional or priming melody-based music generation [14], global control of performance style [14, 25] or fine-grained control of music through learned high-level features [23, 26] and descriptions [19]. Our model is close to the melody-conditioned transformer autoencoder [14], but introduces modifications for the task of score-based performance rendering with style control.

Symbolic Music Encoding: The simplest way to encode symbolic music is a MIDI-like encoding with note-on, note-off, and time-shift events [13, 18]. REMI [15], REMI+ [19], and Compound Word [16] replace position shifts with absolute bar, position, and beat tempo tokens. OctupleMIDI [20] shortens sequence lengths by stacking note attributes into tuples of 8 tokens. For expressive music performance rendering, it is common to mix real, categorical and pianoroll-based score and performance features parsed from MusicXML and MIDI files [8, 9, 11, 27]. Transformers work well with tokenized data [12, 28, 29]. Inspired by OctupleMIDI, we design a tuple-like token encoding that naturally fits aligned score-performance data.

3. DATA ENCODING

3.1 Score and Performance Data Matching

Expressive music performance rendering models require datasets of aligned score and performance music [5, 30]. In this work, we consider piano music performances in MIDI format and use the following data preparation pipeline. First, we compute alignments using Nakamura's alignment tool [31]. The alignments may contain errors, such as alignment holes or close performance notes aligned with distant and unrelated score notes. Following the literature [8, 32], we revise the alignments and filter out notes that deviate from the local performance tempo. After the cleanup, we omit performances with less than 80% aligned notes. Finally, to achieve a perfect match, we remove extra performed notes and interpolate missing notes using the local performance tempos and dynamics, since taking only matched notes and discarding score notes can result in the removal of important chord and bar information.

3.2 SPMuple Encoding

We introduce the Score-Performance Music tuple (SPMuple), a tokenized representation for aligned symbolic score and performance music. It encodes performed notes using tuples of 8 score and 4 performance tokens.

Score Tokens: a set of features extracted from the score MIDI. **Pitch** is a MIDI pitch number in the range 21 to 108. **Duration** is a score note value, encoded by 128 tokens with high and low resolution tokens for short and long durations, respectively [20, 33]. **Bar** is an index of the musical bar to which the note refers, ranging from 0 to the maximum bar in the data. **Position** is the position of the note in the bar, one of 128 tokens with 64th note resolution. **TimeSignature** is the time signature of the beat containing the note, a set of 22 tokens for 2nd, 4th, and 8th note beat lengths, with a maximum bar length of 2 whole notes for 2nd note, and 1.5 for 4th and 8th note. **OnsetShift** is the positional interval between the current and previous note onsets (chords). **NotesInOnset** and **PositionInOnset** are the number of notes and the index of the note in the onset, ranging from 1 to 12, notes are ordered by pitch.

Performance Tokens: a set of performance features extracted from the performance MIDI and processed using the aligned score note features. **Velocity** is a MIDI velocity from 1 to 127. **Tempo** is the performance tempo at the bar, beat or onset level, encoded by a geometric sequence of 121 tokens for beats per minute tempos from 15 to 480. **RelOnsetDeviation** models the exact timing of the note, encoded as the ratio of the absolute note-onset position deviation to the inter-onset interval scaled by the local onset tempo using 161 tokens for values in the range -2 to 2. **RelPerformedDuration** is an articulation of the performed note, computed as the ratio of the performed duration to the score duration, scaled by the local onset tempo, and encoded by 121 tokens for logarithmically distributed values between 0.1 and 3.

The score and performance token sequences are sorted by score note start position, pitch and duration.

3.3 Local Tempo

Inset-onset tempos are noisy and have very high variance, while beat and bar tempos are smoother but still fluctuate at beat/bar boundaries, which can lead to degraded musical experience [34–36]. We design a smooth alternative, local onset tempos, weighted with respect to previous onsets in the local onset time window.

Let $\{\mathrm{IOI}_i^s\}$ and $\{\mathrm{IOI}_i^p\}$ be the sets of score and performance inter-onset intervals between the onset o and N preceding onsets o_i in the time window W. The weights w_i^o for inter-onset tempos $\frac{\mathrm{IOI}_i^s}{\mathrm{IOI}_i^p}$ are computed as:

$$w_i^o = 1 - \frac{\mathrm{IOI}_i^p}{\max_j\{\mathrm{IOI}_j^p\} + 10^{-2}} \tag{1}$$

The weights give more attention to the closest preceding onsets, but still consider the more distant onsets to smooth the local tempo. Based on the decoding quality, we set the time window length W to 8s as the optimal one. In

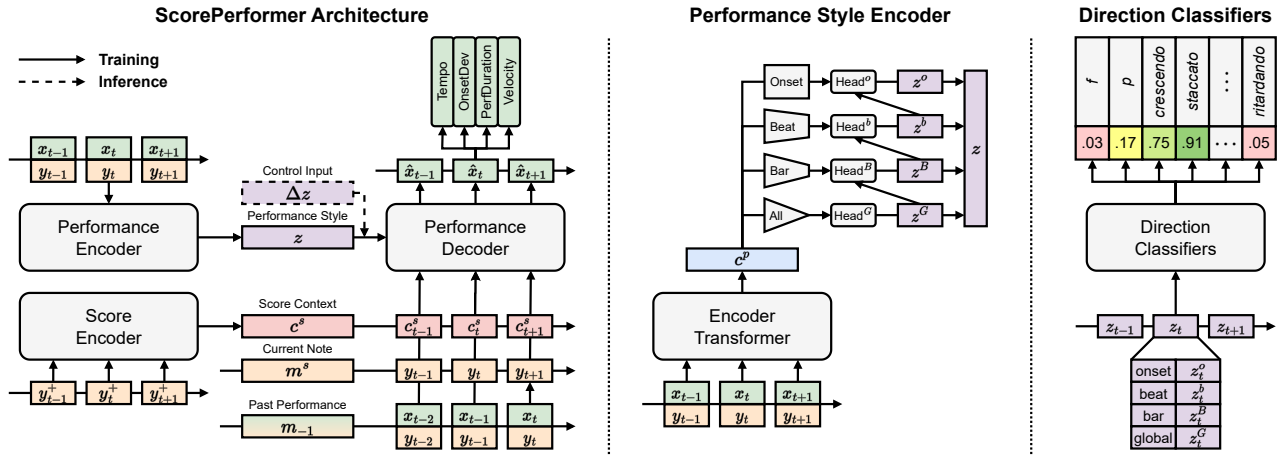

Figure 1. The overall architecture of ScorePerformer, hierarchical style encoding heads and direction classifiers.

addition to the window W, we filter out the nearest onsets with $\text{IOI}_i^p < 0.5$ to reduce the effect of immediate tempo changes, and take at least $N_{\min} = 8$ past onsets with any IOI^p to have enough points for smoothing ($N \geq N_{\min}$).

4. MODEL

With a focus on hierarchical performance style control and efficient training on tokenized sequences, we present **ScorePerformer**, an encoder-decoder model that combines transformers [12] and maximum mean discrepancy variational autoencoders (MMD-VAE) [21,22] for controllable expressive rendering of piano performances for written scores. The model is illustrated in Figure 1.

4.1 Model Architecture

Score Encoder is an encoder transformer that computes a contextual representation of the written music. It maps a score note sequence $y^+ \in \mathbb{N}^{N\times 10}$ (score tokens y + score tempos and velocities) to note embeddings $c^s \in \mathbb{R}^{L\times D}$.

Performance Encoder is an encoder transformer that computes performance style representations at different levels of the musical hierarchy. It takes a sequence of music tuples of score and performance tokens $m = [y, x]$, $y \in \mathbb{N}^{N\times 8}, x \in \mathbb{N}^{N\times 4}$, and outputs performance context embeddings $c^p \in \mathbb{R}^{N\times D}$. The embeddings are grouped and averaged over the entire sequence, bars, beats, and onsets, and iteratively passed through conditional linear layers to compute global, bar, beat, and onset latents z^G, z^B, z^b, and z^o. With the idea of learning missing lower-level details hierarchically, at each step t the latent z_t^* depends on the context c_t^p and all higher-level latents containing the note, e.g. $z_t^b = f_\phi^b(c_t^p, z_t^G, z_t^B)$. All note latents are stacked to produce note-level style embeddings $z \in \mathbb{R}^{N\times D_z}$.

The latent spaces are fit into the Gaussian distribution using a maximum mean discrepancy objective:

$$\mathcal{L}_{\text{MMD}}(p\|q) = \mathbb{E}_{p(z),p(z')}[k(z,z')] + \mathbb{E}_{q(z),q(z')}[k(z,z')] - 2\mathbb{E}_{p(z),q(z')}[k(z,z')], \quad (2)$$

where $k(z,z') = e^{-\frac{\|z-z'\|^2}{2\sigma^2}}$ is a Gaussian kernel.

We use MMD-VAE [21, 22] to solve issues with posterior collapse and latent space holes [37] common to conventional variational autoencoders [38], especially, when trained on sequential data [39].

Performance Decoder is a decoder transformer that renders performance by sequentially predicting performance tokens x_t for score note tokens y_t. The input token sequence combines two sequences: 1) a sequence $m^s = y$ with the current score notes to be rendered; 2) a sequence $m_{-1} = [y_{-1}, x_{-1}]$ shifted one step into the past score y_{-1} and rendered performance tokens x_{-1} describing the past performance history. To reuse the SPMuple token embedder, the first sequence is extended with the masked performance tokens. The two sequence embeddings are concatenated with the score context c^s and passed to the transformer layers together with the style embeddings z.

We use style-adaptive layer normalization (SALN) [40] and pass style embeddings z to the decoder's layer normalization layers, rather than concatenating the style and input token embeddings, to increase the focus on the performance style at each transformer layer.

The performance decoder minimizes the negative log-likelihood for the sequence of performance tokens x:

$$\mathcal{L}_{\text{perf}} = -\sum_{t=1}^{N} \log p_\theta(x_t | x_{<t}, y_{\leq t}, c_{\leq t}, z_{\leq t}) \quad (3)$$

4.2 Transformer Modifications

Discrete+Continuous Tokens: Discrete musical tokens do not explicitly encode the absolute and relative information about note attributes, e.g. that pitches C2, C3, and C4 differ by an octave, or that velocity 80 is louder than 60. We mix discrete and continuous tokens by summing learned discrete token embeddings with delta embeddings provided by a learned nonlinear mapping of the real values associated with tokens to the token embedding dimensions.

Relative Attention: We use the learned ALiBi relative positional bias [41] in the decoder and the learned bidirectional symmetric bias [42] in the encoder for efficient interpolation to sequence lengths not seen during training.

Other: We use single key-value attention heads [43] to speed up decoding, SwiGLU activation [44,45] in feedforward layers, reuse token embedding weights between the encoders and decoder since they share token vocabularies, and tie input and output embeddings in the decoder [45].

4.3 Performance Direction Classifiers

We provide an intuitive interpretation of the learned style embedding space by training performance direction classifiers on the learned note style embeddings. We extract performance direction markings from MusicXML files and associate score notes with performance direction labels where they are present. We train classifiers for **dynamics** (degrees of *piano* and *forte*), **dynamic changes** (*crescendo* and *diminuendo*), **tempo** (*adagio*, *largo*, etc.), **tempo changes** (*accelerando*, *ritardando*, etc.) and **note articulation** binary classes (*staccato*, *fermata*, etc.).

Classifiers take as input the combined note-level performance style embeddings $z = [z^G, z^B, z^b, z^o]$ and output the probabilities of directions being performed in a given performance context. The module minimizes the sum of cross entropy losses for K classifiers with C_k classes each:

$$\mathcal{L}_{\text{clf}} = \sum_{k=1}^{K} \mathcal{L}_{\text{clf}}^{k} = -\frac{1}{N} \sum_{k=1}^{K} \sum_{t=1}^{N} \sum_{c=1}^{C_k} d_{t,c}^{k} \log(\hat{d}_{t,c}^{k}), \quad (4)$$

where $d_{t,c}^{k}$ and $\hat{d}_{t,c}^{k}$ are true and predicted labels for direction c of the classifier k at step t.

Given the smoothness of the learned latent space, the differences between embeddings with high and low classification scores for a given marking may provide a direction in the latent space to move the generation toward the marking. We compute and use mean per-marking delta embeddings to control performance rendering. Since markings are related to defined musical concepts, we can map natural language commands, such as *"play more piano here"* or *"switch to largo"*, to quantitative model control inputs.

4.4 Training and Inference

The total loss minimized by the model during training is:

$$\mathcal{L} = \mathcal{L}_{\text{perf}} + \mathcal{L}_{\text{MMD}} + \mathcal{L}_{\text{clf}} \quad (5)$$

To avoid overfitting of the decoder to lower-level performance embeddings during training, we drop bar, beat, and onset embeddings with probabilities of 0.1, 0.2, and 0.4, respectively. The embeddings are dropped inclusively, i.e. if the bar latent is dropped out, all beat and onset latents are also dropped. Additionally, the classifiers are trained on detached style embeddings z, as we found the model to overfit the unbalanced direction markings labels.

During inference, the sampled or modified reference performance embeddings can be used to control the rendering of the music performance. Based on the learned style spaces, the control can range from high-level global to low-level onset. The extracted performance direction delta embeddings can be used to provide intuitive, command-driven performance manipulation. The model supports real-time inference on the CPU for use in interactive applications.

5. EXPERIMENTS

Datasets: For all experiments, we use the ASAP dataset of matched piano scores and performances [46], preprocessed as described in Section 3.1. The prepared dataset represents 212 musical compositions by 15 composers with a total of 937 performances, 79 hours of performed music. The data is divided into training and evaluation sets with an approximate ratio of 9:1 for the number of performances in the entire dataset and for each composer.

Implementation: The SPMuple data encoding is implemented using `miditok`'s [33] MIDI tokenizer interface. The encoders and decoders in all experiments have a hidden dimension of 256, 4 layers, and 4 attention heads, except for the score encoder, which has 2 layers. The token embedding dimension is set to 128 for each token type, the projected embedding dimension for input embeddings is set to 256. The global, bar, beat, and onset latent dimensions are set to 32, 20, 8, and 4, respectively.

Training: The maximum sequence length during training is set to 256 tokens. To regularize the model and artificially increase the variety of data, we augment the data with sampled pitch shifts (up to ± 3 semitones) and velocity shifts (up to ± 12 MIDI values). In addition, we randomly replace real performances with deadpan performances with a probability of 25% to allow the model to learn the style of both expressive and inexpressive music. We use the ADAM optimizer [47] with an initial learning rate of $2 \cdot 10^{-4}$, decaying by 0.995 after each epoch. Models are trained for 70,000 iterations with batch size 128.

Evaluation: We conduct three sets of experiments: 1) evaluation of the designed data encoding and different local tempo calculation functions; 2) comparison of different latent style hierarchies and their impact on performance rendering; 3) an ablation study on the model architecture design. For the metrics, we use Pearson correlation [9, 11, 48] and mean absolute error for performance features: inter-onset intervals (IOI), absolute onset deviations (OD), performed note durations (PD), and velocity (Vel). We generate 3 samples for each performance in the evaluation set and compute and average the metrics between the ground truth and the generated performances, decoded to MIDI. The errors are measured in seconds, except for velocities, which are measured in MIDI velocity values. After the objective evaluation, we analyze the generation and control capabilities of the designed ScorePerformer model.

6. EVALUATION

6.1 Encoding and Local Tempos

The tokenized representation of performance is not lossless, since some information is lost during feature quantization. We evaluate the decoding quality and performance of ScorePerformer on sequences encoded using SPMuple with different local tempo functions.

Table 1 shows the evaluation results. The local window onset tempo function (Section 3.3) shows the least degradation in decoding quality for inter-onset intervals and onset deviations. It captures local tempo changes and note

	Decoded						**Generated,** $\Delta z = 0$							
	Error ↓			Correlation ↑			Error ↓				Correlation ↑			
Tempo	IOI	OD	PD	IOI	OD	PD	IOI	OD	PD	Vel	IOI	OD	PD	Vel
Bar	0.092	0.002	0.026	0.770	0.953	0.954	0.140	0.012	0.063	**2.354**	0.650	0.361	0.837	0.940
Beat	0.084	0.002	0.027	0.836	0.971	0.958	0.116	0.009	0.066	2.627	0.727	0.406	0.854	0.932
Onset	**0.019**	**0.001**	**0.006**	0.921	0.977	**0.982**	0.124	0.011	0.056	2.856	0.709	0.339	0.890	0.932
Window	0.028	**0.001**	0.011	**0.963**	**0.985**	0.979	**0.090**	**0.008**	**0.048**	2.583	**0.901**	**0.538**	**0.907**	**0.943**

Table 1. Encoding evaluation on decoded performances and performances generated with unaltered style embeddings from the performance encoder. IOI – inter-onset interval, OD – onset deviation, PD – performed duration, Vel – velocity.

G	B	b	o	z	IOI	OD	PD	Vel
32	**20**	**8**	**4**	**64**	**0.901**	**0.538**	**0.907**	**0.943**
32	20	12	✗	64	0.464	0.194	0.739	0.861
32	32	✗	✗	64	0.417	0.067	0.722	0.812
64	✗	✗	✗	64	0.327	0.066	0.658	0.576
✗	32	✗	✗	32	0.410	0.069	0.702	0.792
✗	✗	12	✗	12	0.384	0.066	0.711	0.767
✗	✗	✗	4	4	0.590	0.063	0.735	0.748
32	20	8	✗	60	0.410	0.065	0.764	0.847
32	20	✗	4	56	0.842	0.224	0.881	0.857
32	✗	8	4	44	0.863	0.386	0.886	0.913
✗	20	8	4	32	0.890	0.485	0.904	0.939

Table 2. Correlation with ground truth performances for samples generated by models trained with different combinations of latent hierarchies. G – global, B – bar, b – beat, o – onset, and z – total latent dimensions.

timing more efficiently than bar, beat and onset tempos. These findings are supported by the generation results. The model trained with local window tempo tokens renders samples with smaller errors and closer to the ground truth than the models trained with bar, beat, or onset tempo tokens. In particular, it shows more consistency in modeling local tempo changes and note timing. For future work, the encoding could be further improved by incorporating pedals, an essential element of piano performance [49].

6.2 Style Embedding Hierarchies

Table 2 shows the impact of different learned style embedding hierarchy combinations in ScorePerformer on the quality of performance rendering. Replacing lower-level latents with higher-level ones, using only a single level, or omitting any level of the hierarchy leads to a decrease in quality for all musical features. The lower-level onset latents account for most of the variation in performance features, while the higher-level latents provide the missing performance timing, articulation, and dynamics information at the beat, bar, and global levels. The results suggest that a hierarchical style representation is advantageous for modeling global and local changes in music performance. The search for an optimal configuration of latent dimensions is beyond the scope of this study.

	IOI	OD	PD	Vel
ScorePerformer	**0.901**	**0.538**	**0.907**	0.943
w/o Score Encoder	0.885	0.526	0.889	**0.951**
w/o input seq. m^s	0.844	0.422	0.895	0.925
w/o SALN	0.871	0.469	0.920	0.930
w/o in-out emb. tie	**0.901**	0.459	0.873	**0.951**
w/o Continuous Tokens	0.576	0.116	0.747	0.561

Table 3. Evaluation of model configurations using the correlation between ground truth and generated performances.

6.3 Ablation Study

The ablation study on the ScorePerformer model is summarized in Table 3. Removing any of the proposed design choices degrades the quality for all features in almost all cases. The score encoder adds a local future score context to the decoder and contributes to a slight quality improvement. The same is true for the additional decoder input sequence m^s, which explicitly highlights the currently rendered score notes. Without style-adaptive layer normalization or input-output embedding weight sharing, the correlation for timing features decreases. The most noticeable quality degradation occurs after using only discrete tokens without continuous input tokens, demonstrating the positive impact of value-aware inputs on model predictions.

6.4 Performance Embeddings Analysis

We explore the learned performance style spaces using the trained performance direction marking classifiers. We take the style embeddings z for note onsets in the dataset and project them into two dimensions using principal component analysis [50]. Figure 3 shows the projected embeddings labeled by the selected dynamics, tempo, and articulation markings classifiers and their ground truth labels. We can see the gradient moving from the light colors (high probabilities) to the darker colors (low probabilities). Despite the class imbalances and low representation of some labels in the dataset, the positive classifier predictions match the areas of the ground truth labels shown in the right plots for each marking. This suggests that the vectors for moving the performance toward the markings exist in the original latent style space and can be used to attempt to control the performance rendering through the model.

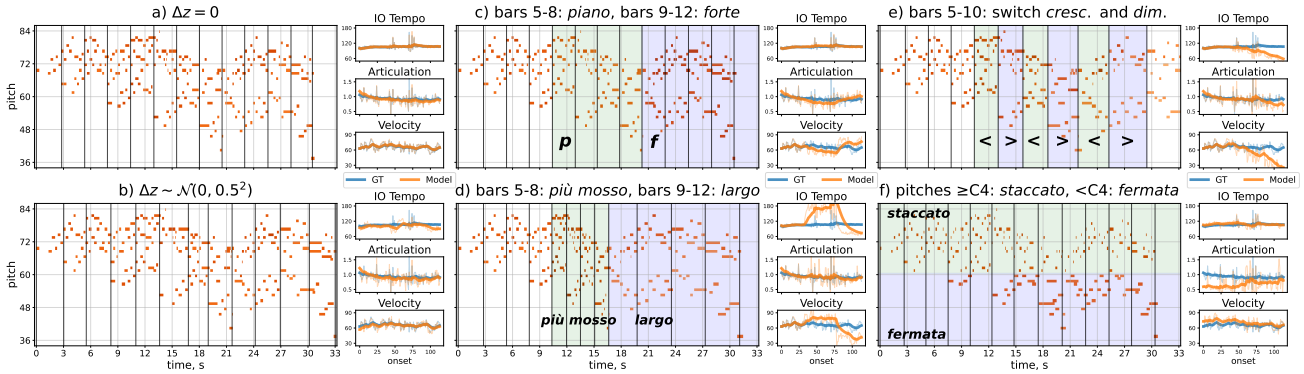

Figure 2. Pianorolls and performance features (inter-onset tempo, articulation, and velocity) for the first 12 musical bars of Bach's "Prelude and Fugue No.19", rendered by ScorePerformer with unconditional or conditional style control. The title of each plot indicates the form of the control input. Colored areas highlight the regions with the applied control.

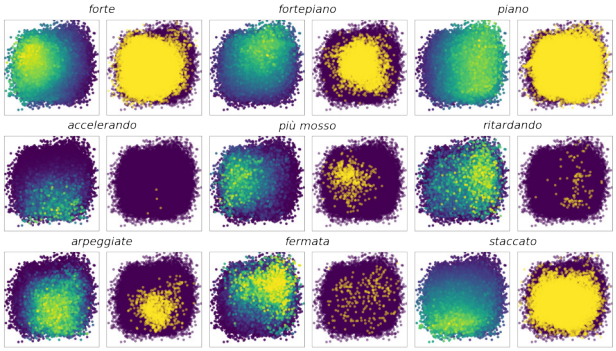

Figure 3. Projected style embeddings classified by chosen direction marking classifiers. The left and right plots for each marking highlight predicted and ground truth labels.

The direction classifiers can also be used to analyze performance practices. For example, take all performance contexts with a given notated performance direction marking and sort them by the classification scores using the associated direction classifier. Further analysis of the score contexts can provide insight into the reasons why musicians follow or interpret differently certain markings.

6.5 Performance Rendering Control

For performance rendering control, we add control embeddings Δz to the encoded style embeddings and pass them to the decoder. We analyze both uncontrolled generation with sampled control embeddings and direction-based control using the computed delta latents for markings.

Figure 2 shows examples of music performance rendering for a composition from the evaluation set. The sample (a) shows the successful reconstruction of the performance variations from the encoded style embeddings. When generated using sampled delta latents (b), the added noise is transferred to higher variations in tempo than in articulation and dynamics. In our observations, small amounts of delta noise can result in both pleasant and diverse samples.

From the evaluation of the performance direction based control, we can see that in most cases the model follows the musical meaning of the marking. Example (c) shows that *piano* and *forte* delta embeddings lead to the expected decrease and increase in dynamics. The *più mosso* (more movement, faster) and *largo* (slowly and broadly) in the example (d) lead to the expected changes in tempo, articulation and dynamics. An interesting behaviour can be found in example (e), where the model values one marking over the other. During the alternation of *crescendo* and *diminuendo*, the model follows *diminuendo* more and falls on the path of slow and quiet performance. The last example (f) shows that the control can also be applied effectively to individual notes. As the definitions suggest, the *staccato* on higher pitched notes makes them more abrupt, and the *fermata* on other notes holds the notes a bit longer.

Despite the positive examples of piano performance rendering control, the model has some limitations. The proposed marking delta embeddings encode the highest learned deviations between performance styles and lead to immediate changes in performance, which can sound unnatural. One solution is to scale or interpolate the control inputs for smoother performance changes. Another issue to be addressed is disentangling the learned latent space across direction classes for a more controllable generation. Finally, the study was limited by low performance variation for some markings and compositions in the dataset. We believe that the proposed approach has a high potential for both analytical and musical creativity applications that could be fulfilled with orders of magnitude larger datasets.

7. CONCLUSION

We presented ScorePerformer, an encoder-decoder transformer with hierarchical MMD-VAE style encoding heads for fine-grained controllable expressive rendering of piano music performances. We also introduced performance direction classifiers, trained on performance style embeddings, to map notated direction markings and natural language inputs to model control inputs. Evaluation showed that the model captures performance style variations and follows control intents. Future work will focus on improving the diversity of training data to enable large-scale analysis, and may include in-depth subjective evaluation of the proposed and existing performance rendering models.

8. ACKNOWLEDGEMENTS

We thank Dmitry Yarotsky for his valuable comments during the model development and experimentation. We thank the anonymous reviewers for their critical feedback, which allowed us to improve the paper. The experiments were performed on the Zhores computing cluster [51].

9. REFERENCES

[1] C. Palmer, "Music performance," *Annual review of psychology*, vol. 48, no. 1, pp. 115–138, 1997.

[2] J. Rink, *Musical Performance: A Guide to Understanding*. Cambridge University Press, 2002.

[3] S. Ji, J. Luo, and X. Yang, "A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions," *arXiv preprint arXiv:2011.06801*, 2020.

[4] C. Hernandez-Olivan, J. Hernandez-Olivan, and J. R. Beltran, "A Survey on Artificial Intelligence for Music Generation: Agents, Domains and Perspectives," *arXiv preprint arXiv:2210.13944*, 2022.

[5] A. Kirke and E. R. Miranda, *Guide to Computing for Expressive Music Performance*. Springer, 2013.

[6] C. E. Cancino-Chacón, M. Grachten, W. Goebl, and G. Widmer, "Computational Models of Expressive Music Performance: A Comprehensive and Critical Review," *Frontiers in Digital Humanities*, vol. 5, p. 25, 2018.

[7] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019, pp. 3060–3070.

[8] D. Jeong, T. Kwon, Y. Kim, K. Lee, and J. Nam, "VirtuosoNet: A Hierarchical RNN-based System for Modeling Expressive Piano Performance," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 908–915.

[9] A. Maezawa, K. Yamamoto, and T. Fujishima, "Rendering Music Performance With Interpretation Variations Using Conditional Variational RNN," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 855–861.

[10] H. H. Tan, Y.-J. Luo, and D. Herremans, "Generative modelling for controllable audio synthesis of expressive piano performance," *arXiv preprint arXiv:2006.09833*, 2020.

[11] S. Rhyu, S. Kim, and K. Lee, "Sketching the Expression: Flexible Rendering of Expressive Piano Performance with Self-Supervised Learning," *arXiv preprint arXiv:2208.14867*, 2022.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008.

[13] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "Music Transformer: Generating Music with Long-Term Structure," *arXiv preprint arXiv:1809.04281*, 2018.

[14] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding Musical Style with Transformer Autoencoders," *arXiv preprint arXiv:1912.05537*, 2019.

[15] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions," *arXiv preprint arXiv:2002.00212*, 2020.

[16] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 178–186.

[17] B. Yu, P. Lu, R. Wang, W. Hu, X. Tan, W. Ye, S. Zhang, T. Qin, and T.-Y. Liu, "Museformer: Transformer with Fine-and Coarse-Grained Attention for Music Generation," *arXiv preprint arXiv:2210.10349*, 2022.

[18] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, vol. 32, pp. 955–967, 2020.

[19] D. von Rütte, L. Biggio, Y. Kilcher, and T. Hoffman, "FIGARO: Generating Symbolic Music with Fine-Grained Artistic Control," *arXiv preprint arXiv:2201.10936*, 2022.

[20] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training," *arXiv preprint arXiv:2106.05630*, 2021.

[21] S. Zhao, J. Song, and S. Ermon, "InfoVAE: Information Maximizing Variational Autoencoders," *arXiv preprint arXiv:1706.02262*, 2017.

[22] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, "A Kernel Method for the Two-Sample-Problem," in *Advances in Neural Information Processing Systems*, vol. 19. MIT Press, 2006, pp. 513—-520.

[23] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-Song and Fine-Grained Piano Music Style Transfer with One Transformer VAE," *arXiv preprint arXiv:2105.04090*, 2021.

[24] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2018, pp. 4364–4373.

[25] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer," *arXiv preprint arXiv:1809.07600*, 2018.

[26] H. H. Tan and D. Herremans, "Music FaderNets: Controllable Music Generation Based On High-Level Features via Low-Level Feature Modelling," *arXiv preprint arXiv:2007.15474*, 2020.

[27] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Score and performance features for rendering expressive music performances," in *Music Encoding Conference*. Music Encoding Initiative Vienna, Austria, 2019, pp. 1–6.

[28] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*, 2020.

[29] Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, O. Teboul, D. Grangier, M. Tagliasacchi, and N. Zeghidour, "AudioLM: a Language Modeling Approach to Audio Generation," *arXiv preprint arXiv:2209.03143*, 2022.

[30] C. E. Cancino-Chacón, "Computational Modeling of Expressive Music Performance with Linear and Nonlinear Basis Function Models," Ph.D. dissertation, Johannes Kepler University Linz, Austria, December 2018.

[31] E. Nakamura, K. Yoshii, and H. Katayose, "Performance Error Detection and Post-Processing for Fast and Accurate Symbolic Music Alignment," in *International Society for Music Information Retrieval Conference*, 2017.

[32] G. G. Xia, "Expressive collaborative music performance via machine learning," Ph.D. dissertation, Carnegie Mellon University, August 2016.

[33] N. Fradet, J.-P. Briot, F. Chhel, A. El Fallah-Seghrouchni, and N. Gutowski, "MidiTok: A Python package for MIDI file tokenization," in *22nd International Society for Music Information Retrieval Conference*, 2021.

[34] S. Dixon, W. Goebl, and E. Cambouropoulos, "Perceptual Smoothness of Tempo in Expressively Performed Music," *Music Perception*, vol. 23, no. 3, pp. 195–214, 2006.

[35] B. H. Repp, "On Determining the Basic Tempo of an Expressive Music Performance," *Psychology of Music*, vol. 22, no. 2, pp. 157–167, 1994.

[36] H. Schreiber, F. Zalkow, and M. Müller, "Modeling and Estimating Local Tempo: A Case Study on Chopin's Mazurkas," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020, pp. 773–779.

[37] J. Lucas, G. Tucker, R. B. Grosse, and M. Norouzi, "Understanding Posterior Collapse in Generative Latent Variable Models," in *Deep Generative Models for Highly Structured Data, ICLR 2019 Workshop*, 2019.

[38] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[39] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio, "Generating Sentences from a Continuous Space," in *Conference on Computational Natural Language Learning*, 2015.

[40] D. Min, D. B. Lee, E. Yang, and S. J. Hwang, "Meta-StyleSpeech: Multi-Speaker Adaptive Text-to-Speech Generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 7748–7759.

[41] O. Press, N. A. Smith, and M. Lewis, "Train short, test long: Attention with linear biases enables input length extrapolation," *arXiv preprint arXiv:2108.12409*, 2021.

[42] M. Lee, K. Han, and M. C. Shin, "LittleBird: Efficient Faster & Longer Transformer for Question Answering," *arXiv preprint arXiv:2210.11870*, 2022.

[43] N. Shazeer, "Fast Transformer Decoding: One Write-Head is All You Need," *arXiv preprint arXiv:1911.02150*, 2019.

[44] ——, "GLU Variants Improve Transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[45] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "PaLM: Scaling Language Modeling with Pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[46] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: a Dataset of Aligned Scores and Performances for Piano Transcription," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020, pp. 534–541.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[48] C. E. Cancino-Chacón, T. Gadermaier, G. Widmer, and M. Grachten, "An evaluation of linear and non-linear models of expressive dynamics in classical piano and symphonic music," *Machine Learning*, vol. 106, pp. 887–909, 2017.

[49] S. P. Rosenblum, "Pedaling the Piano: A Brief Survey from the Eighteenth Century to the Present," *Performance Practice Review*, vol. 6, no. 2, p. 8, 1993.

[50] M. E. Tipping and C. M. Bishop, "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 61, no. 3, pp. 611–622, 1999.

[51] I. Zacharov, R. Arslanov, M. Gunin, D. Stefonishin, A. Bykov, S. Pavlov, O. Panarin, A. Maliutin, S. Rykovanov, and M. Fedorov, ""Zhores"—Petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology," *Open Engineering*, vol. 9, no. 1, pp. 512–520, 2019.