RENDERING MUSIC PERFORMANCE WITH INTERPRETATION VARIATIONS USING CONDITIONAL VARIATIONAL RNN

Akira Maezawa Kazuhiko Yamamoto Takuya Fujishima
Yamaha Corporation

ABSTRACT

Capturing and generating a wide variety of musical expression is important in music performance rendering, but current methods fail to model such a variation. This paper presents a music performance rendering method that explicitly models the variability in interpretations for a given piece of music. Conditional variational recurrent neural network is used to jointly train, conditioned on the music score, an encoder from a music performance to a latent representation of interpretation and a decoder from the latent interpretation back to the music performance. Evaluation demonstrates the method is capable of predicting and generating an expressive performance, and that the decoder learns a latent space of musical interpretation that is consistent with human perception of interpretation.

1. INTRODUCTION

Music performance rendering is the task of generating a human-like performance data from a piano music score. That is, for each note in a given music score, it generates a set of expressive performance parameters such as the onset timing, the duration and the velocity. It is an important task in music production, allowing a composer to generate human-like piano part of a new composition, or a musician to listen to a convincing preview of a digital sheet music. It is not only important to generate a convincing performance, but also to allow humans to interact in the generation. For example, it is useful to be able to adjust expressive parameters, or, like how a musician might ask to another musician, feed a reference performance snippet, in expression of which the system should perform.

Capturing and generating a wide variety of musical expression are important in music performance rendering, but current methods are limited in such capabilities. For example, most methods permit control over concrete musical concepts like the average tempo and the average velocity [2], but cannot manipulate abstract musical concepts that cannot be labeled, such as liveliness within a performance. It is also not possible to feed a short reference performance snippet and ask the system to play with that particular musical musical interpretation.

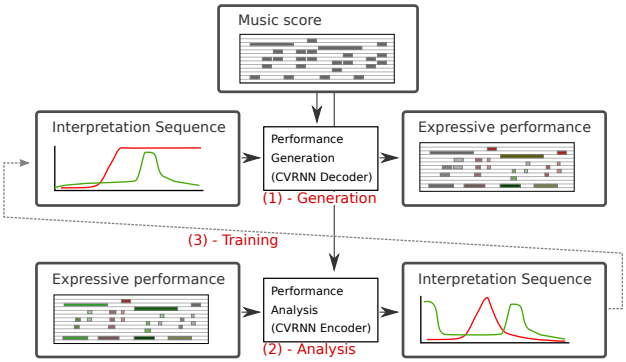

Figure 1. The overview of our method.

These limitations are rooted in the attempt to directly map between the music score and expressive performance [4]. Such a model entwines musical interpretation and its execution, but a human performer presumably decouples *musical intent* and its *execution* given the score. For example, when playing a piece multiple times, a musician might play lively the first time but calmly on the second. Given the intent, however, its execution is consistent based on musical contexts, such as whether a note is a melody or if a particular measure is a beginning of a new phrase. For example, a lively playing might be executed, almost second nature, as shorter chords and louder melody, or calm playing as softer and more broken chords.

To address this problem, we present a music performance rendering method that explicitly decouples intent and its execution based on a music score. We achieve this by jointly training a performance decoder (performance renderer) and a performance encoder (performance analyzer) that are conditioned on (1) a music score, and (2) an **interpretation sequence**, a latent low-dimensional sequence representation that expresses the underlying musical intent. It may be either generated automatically or manipulated coherently. Thus, it is possible to generate different interpretations to a score by either modifying the interpretation sequence or encoding a performance snippet as an initial value for generating the interpretation sequence.

The conceptual overview is shown in Figure 1. Our method, given a music score and an interpretation sequence, generates an expressive performance for the given music (denoted (1)). Conversely, it can encode an expressive performance into the interpretation sequence (denoted (2)). To achieve such a capability, the encoder and the de-

coder are jointly trained by encoding a human performance and trying to recover it with the decoder (denoted by path (3)).

Our contributions are as follows: (1) we propose a new machine learning model, CVRNN, which extends variational recurrent neural network (VRNN [6]) to accept position-dependent conditional inputs; (2) we apply convolutional recurrent network (CRNN) to extract features from the music score; (3) we apply these two models and present a first deep generative model that is simultaneously capable of analyzing and generating an expressive music performance for a given music score with adjustable musical interpretation.

We invite the readers to check out audio demonstrations at `https://sites.google.com/view/cvrnn-performance-render`.

2. RELATED WORK

Music performance rendering is the task of generating an expressive performance from a music score [4]. The research focuses mostly on generating a sequence of expressive onset timings, durations and velocities for a piano piece. Unlike the generation of improvisation [20], performance rendering needs to understand the musical contexts of the given music score, and generate a natural performance constrained by the score.

Previous studies formulate performance rendering as a prediction task of expressive parameters based on features extracted from the music score, such as melodic features [22], perceptual features [3], or neighboring contexts [9, 18]. Then, expressive performance to a new music score can be predicted using methods like a hidden Markov model [12], a decision tree [18], or a dynamic Bayesian network [8]. More recently, deep learning has been incorporated to better extract the features or predict the performance. For example, deep neural networks have been shown to be effective for dynamics prediction [21], tempo prediction [15], and piano performance generation [11, 17].

The capability to adjust the generated performance is important, but current methods cannot capture the abstract variations in playing. For example, the tempo or dynamics [1, 7] can be adjusted, but controlling a more abstract musical idea remains difficult. It is possible to manipulate the performance by mixing the parameters of performance renderers trained on multiple corpora [2], but it fundamentally cannot identify commonalities and differences among a set of corpora.

3. METHOD

Our method generates an expressive piano performance data to a given piano music score, and an interpretation sequence that is either automatically generated, manually adjusted, or initialized by a performance snippet. The score contains a sequence of notes (pitch, position and duration), the average tempo, the time signatures, and the key signatures. The interpretation sequence is a sequence of low-dimensional vectors, each element of which is associated with each note in the score. It is an abstract representation of playing style, each vector of which is called the **interpretation vector**. For each note, based on the interpretation vector and musical context inferred from the music score, our method estimates (1) the fine note onset timing, (2) the duration, (3) the note-on velocity, and (4) the tempo fluctuation of human performance.

To generate the performance, we use a deep generative model that has been trained to generate an expressive performance data, conditioned on the music score and the interpretation sequence. To acquire a temporally coherent manifold over the space of interpretation vector, we propose CVRNN, an extension of VRNN that accepts the music score as a conditioning input. Essentially, we jointly train three models: (1) a music score feature extractor that extracts, for each note, relevant features (**music score feature**) from the music score, (2) an encoder that maps a human performance to an interpretation sequence, and (3) a decoder that maps an interpretation sequence to an expressive performance.

Hereon, n indicates the index of a note encountered in the music score, lexicographically sorted by the onset beat position and the pitch. x_n indicates the expressive performance data, c_n indicates the music score feature that has been extracted from the nth note on the score, and z_n indicates the interpretation vector associated with x_n.

3.1 Representation of expressive performance

The generated expressive performance x_n comprises of the note velocity, the note duration, the micro-onset timing deviation and the tempo.

The **note velocity** is an integer between 1 and 127 indicating the MIDI velocity value, encoded as a 128-dimensional one-hot vector.

The **note duration** is an integer between 1 and 800, indicating the duration as $1/100$ths of a beat relative to the current tempo (*e.g.* duration of "42" means 0.42 beats). It is encoded as an 800-dimensional one-hot vector.

The **micro-onset timing** is an integer between -50 and 49, indicating the number of $1/100$ths of a beat elapsed after an ideal onset time. The ideal onset time is computed based on the generated tempo curve up to the current point in the score. It is represented as a 100-dimensional one-hot vector, ith element of which means that the onset is lagging by $(i - 50)/100$ beats.

The **tempo** is an integer indicating the the difference between the current tempo and the tempo written in the music score, in bpm. It is represented as a 100-dimensional one-hot vector, ith element of which means that the output should be faster than the written tempo by $i - 50$ bpm. It is computed by interpolating the music score position and the playback position of the performance using Gaussian regression with a squared exponential kernel with a FWHM of 2 beats.

3.2 Extraction of the music score feature

The music score contains a sequence of notes, average dynamics, average tempi, meters, and key signatures. From

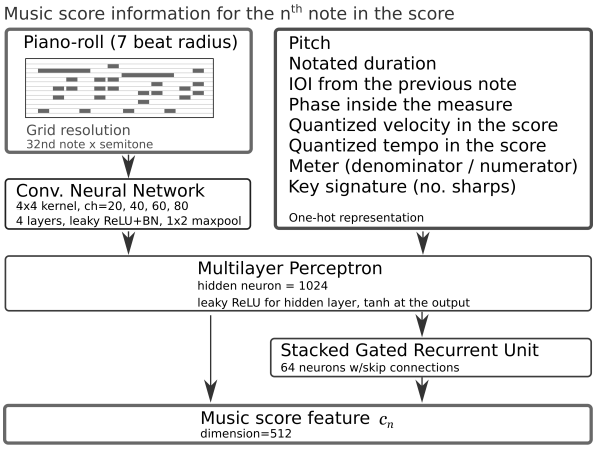

Figure 2. Overview of music score feature extractor for extracting the music score feature.

the music score, we extract the **music score feature** c_n, which represents the nth note and its surrounding context.

To extract the feature, we use the following information: the **key signature** as a 12 dimensional one-hot vector that indicates the number of accidentals; the **time signature** as a tuple of 12 dimensional one-hot vector, indicating the numerator and the denominator; the **pitch** as a 128 dimensional one-hot vector indicating the MIDI note number; the **inter-onset interval** from the previous note as a multiple of 96th note, represented as a 512 dimensional one-hot vector; the **duration** of the note as a multiple of 96th note, represented as a 192 dimensional one-hot vector; the written **tempo** quantized to a multiple of 30 bpm, represented as a 9 dimensional one-hot vector (30-300 bpm); the written **velocity** quantized to a multiple of 10, represented as a 12 dimensional one-hot vector; the **piano-roll** centered about the current position in the score, spanning a radius of 7 beats at a 32nd note resolution (224x128 dimension). The velocity and the tempo are quantized at a low resolution because different music notation software map the dynamics and tempo markings to similar velocity and bpm values, but the exact mapping differ.

Based on these inputs, we extract the feature using a CRNN as shown in Figure 2. First, the piano-roll passes through a CNN. It is comprised of four layers, whose kernel sizes are all (4×4), and channel sizes from the lower to the higher layers are [20, 40, 60, 80]. Each layer is followed by max-pooling with a kernel size of (2×1), batch normalization and leaky ReLU nonlinearity. The output the CNN is concatenated with the remaining one-hot vectors mentioned above. Then, the concatenated vector is passed through a multi-layer perceptron (MLP) with 1024 hidden units and leaky ReLU nonlinearity at the hidden layer. The MLP output passes through a tanh nonlinearity.

Second, the output of the perceptron is embedded to 64 dimensions using a linear layer. The embedded vector is then passed to a RNN consisting of 3 stacks of gated recurrent units (GRU) [5] with 64 neurons each.

Finally, the outputs of the MLP and the GRU stack are concatenated to create the music score feature c_n.

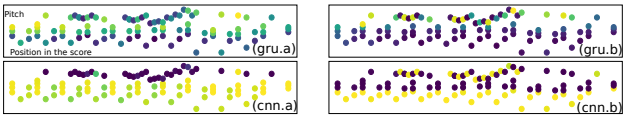

Figure 3. Activation of the most active components of GRU and CNN activations (dark=negative, yellow=positive).

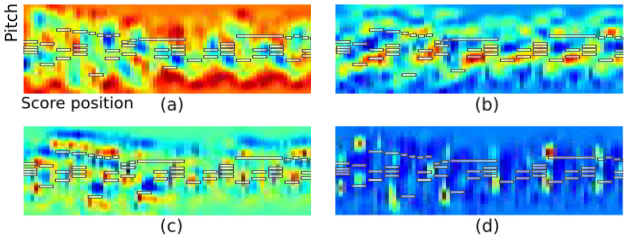

Figure 4. Some feature maps learned from the CNN, overlayed with the piano-roll (blue=negative, red=positive).

3.2.1 Analysis of the music score feature extractor

Here we illustrate some musically relevant concepts that are acquired by the music score feature extractor. We have trained our model using the same dataset used in Section 4, and extracted the music score feature from measures 5–6 of Chopin's Nocturne Op. 9-2 and extracted the GRU and the CNN activations. Some of the highest GRU and CNN activations are shown in Figure 3, showing, for each note, the value of the corresponding activation in different color, positioned at the beat position and the pitch written on the score, *a la* piano-roll. We can see that the GRU acquires concepts like the number of notes stacked below the current point in the score ("gru.a") or top notes ("gru.b"). The CNN extracts longer contextual information such as the melody ("cnn.a"), and the bottom notes ("cnn.b"). Neither the GRU nor the CNN becomes dead: the standard deviation of the activation of the CNN and the GRU are comparable (about 0.5 to 1.0), where about 90 GRU activations have standard deviation of above 0.1, and about 150 for the CNN.

To analyze how these concepts are acquired, we show in Figure 4 the feature maps of the final layers of the CNN, overlayed to the piano-roll. The tendency of the map suggests that the CNN expresses concepts like the register (Fig. 4a), rising arpeggio (b), top and bottom melodic contour (c), or melodic contour (d).

3.3 CVRNN for joint encoding and decoding of the interpretation sequence

We assume that an expressive performance depends on both (1) the musical context, expressed through the music score feature, and (2) a corresponding sequence of low-dimensional music interpretation vectors z_n, dimension of which is set to 5. It is necessary for z_n to learn (1) a manifold that z_n resides in, that explains interpretation reasonably well, and (2) a model of temporal evolution of z_n, so that the generated performance is temporally coher-

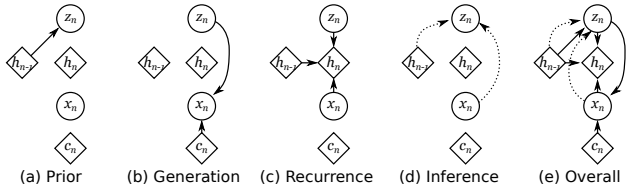

Figure 5. Dependency of the CVRNN model. Solid arrows indicate conditional dependence of the generative process, and dotted arrows indicate conditional dependence of the approximate posterior distribution.

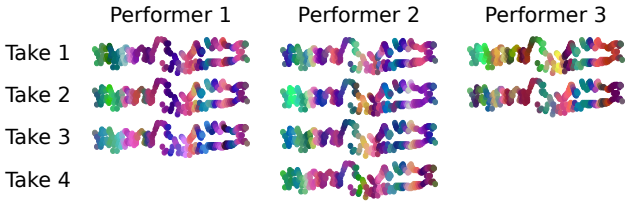

Figure 6. Visualization of the interpretation sequence, for different takes of three professional pianists. Horizontal axis indicates the music score position, vertical axis indicates the pitch, and the color indicates 3D projection of the temporally-smoothed interpretation vector.

ent. We achieve (1) by decoupling the music score and the generative model of z_n, such that z_n learns a repertoire-independent low-dimensional representation of variability of playing. We achieve (2) by basing the temporal evolution of z_n on the state of a RNN h_n.

To meet these requirements, we model the generation process with a CVRNN. There are three key components in a CVRNN, as shown in the dependency diagram in Figure 5: (1) generation of the interpretation sequence z_n given the underlying h_n (called the **prior** distribution), (2) generation of an expressive performance x_n given the interpretation sequence z_n and music score features c_n (called the **generation** distribution), and (3) generation of the interpretation vector z_n given a true human performance x_n (called the **inference** distribution). Finally, the true or the generated expressive performance x_n and the interpretation vector z_n are used to update the underlying h_n, used then to predict the next interpretation vector z_{n+1}.

3.3.1 Generation

We assume the following generative process:

$$p(x_{\leq N}, z_{\leq N}|c_{\leq N}) = \prod_{n=1}^{N} \underbrace{p(x_n|z_n, c_n)}_{\text{generation}} \underbrace{p(z_n|x_{<n}, z_{<n})}_{\text{prior}}. \tag{1}$$

Thus, the data is generated autoregressively by sampling z_n from the prior, generating x_n, and feeding it back to the prior to sample the z_{n+1} for the next note in the score.

For the prior distribution, the previous RNN state h_{n-1} passes through a three-layer densely-connected [10] fully-connected network with 250 units each with a leaky ReLU nonlinearity. By dense connection, we mean that the input to the current layer is a concatenation of the outputs of previous layers. It outputs the mean and log-variance of a Normal distribution $\mu(h_{n-1})$ and $\gamma(h_{n-1})$. Then z_n is sampled as follows:

$$z_n|h_{n-1} \sim \mathcal{N}(\mu(h_{n-1}), \exp(\gamma(h_{n-1}))). \tag{2}$$

From the interpretation vector z_n, we pass it through another feature extractor $\psi(z_n)$ to obtain a feature that describes the current musical interpretation. $\psi(\cdot)$ has the same three-layer architecture as mentioned above.

For the generation of x_n, we use the sampled interpretation vector z_n and the current music context c_n:

$$x_n|z_n, c_n \sim g(c_n, \psi(z_n)), \tag{3}$$

where $g(c, z)$ indicates a Cartesian product of categorical distributions over the one-hot representations of the velocity, the micro-onset timing, the duration and the tempo, parameterized by an output of a three-layer fully-connected network with leaky ReLU nonlinearity for the hidden layers and softmax for the output layer, applied separately for the four output variables.

To recurrently update h_n, we extract a feature vector from x_n by passing it through a network $\phi(\cdot)$ with a same architecture as ψ to obtain $\phi(x_n)$. Then, the underlying state h_n is updated as follows:

$$h_{n+1} = f(h_n, [c_n, \phi(x_n), \psi(z_n)]), \tag{4}$$

where $f(h, x)$ is state update equation of a stacked GRU of three layers, given h as the current state variable and x as the input that has been embedded to 64 dimensions by a fully-connected layer.

Let us elaborate on the modeling assumptions. First, the prior in Eq. 2 is independent of c_n. This forces z_n to express music interpretation abstractly, such that it is decoupled from the musical context, which provides strong hints on how to execute the performance given an interpretation. Second, x_n and h_n depends only indirectly via the bottleneck z_n. This is unlike the original formulation of VRNN where x_n and h_n are directly dependent. We found that such a bottleneck is vitally critical for learning a meaningful representation of z_n; if x_n depends on h_n, then the model simply learns to generate an auto-regressive model of x_n using h_t, almost completely bypassing z_n. Such an error mode occurs because it is much easier for a model to learn to simply predict the next note, instead of having to go through the hassle of trying to explain how it could have varied with a different interpretation.

3.3.2 Inference

In addition to the generative process, we also define an approximate posterior distribution, or the inference distribution, so that variational technique can be used to train the model [14].

We assume the following dependency for the inference distribution:

$$q(z_{\leq N}|x_{\leq N}) = \prod_{n=1}^{N} \underbrace{q(z_n|x_{\leq n}, z_{<n})}_{\text{inference}}. \tag{5}$$

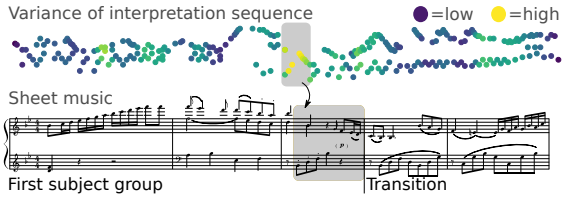

Figure 7. The variance of the interpretation vectors over different performers at each note.

If we assume that z_n is normally distributed and conditionally independent of $z_{<n}$ and $x_{<n}$ given h_{n-1}, it becomes:

$$q(z_n|x_{\leq n}, z_{<n}) = q(z_n|x_n, h_{n-1}) \\ = \mathcal{N}(z_n|\eta(h_{n-1}, \phi(x_n)), \exp(\nu(h_{n-1}, \phi(x_n)))), \quad (6)$$

where $\eta(\cdot)$ and $\nu(\cdot)$ are the outputs of a neural network with the same architecture as $\psi(\cdot)$, that take $\phi(x_n)$ and h_{n-1} as the inputs. The inference is independent of the music score c_n, so that it learns a *repertoire-agnostic* model for inferring z_n given an expressive human playing.

3.3.3 Training

We train the model by minimizing the KL divergence from the approximate posterior to the true posterior. It amounts to the minimizing of the following note-level loss $l_n(\Theta)$ w.r.t. the set of model parameters Θ, accrued over every note in the training data:

$$l_n(\Theta) = \mathbb{E}_{z_n \sim q(z|x_n, h_{n-1})}[\log p(x_n|z_n, c_n)] \\ + \mathrm{KL}(q(z|x_n, h_{n-1})||p(z|h_{n-1})). \quad (7)$$

The expectation is computed using the reparametrization trick [14]. We use truncated backpropagation with truncation length of 30 notes (longer truncation length of 50 notes yielded qualitatively similar outputs). We also augment the data by adding zero-reverting Wiener noise to the tempo curve and randomly transposing each piece by -7 to 7 semitones at every epoch. To minimize the loss, we use Adam [13], with gradient clipping for gradient values above 5.

3.4 Analysis of the learnt manifold of the interpretation vector

We briefly illustrate the essences of musical expression acquired by the interpretation sequence. Figure 6 shows the interpretation sequence of first 20 bars of Mozart's K333 piano sonata, mvt. 1 (from the top to the introduction of second subject group), played by three professional pianists for multiple takes. The figure shows that the interpretation sequence tends to remain similar within each performer, and different among different performers.

We can also show where in the music score has the highest variance of interpretation vector among the nine takes. Figure 7 shows the variance for each note, from which we can see that the maximum variance occurs at the structural boundary from the first subject group to the transition, supporting findings that music expression varies significantly at structural boundaries [19].

4. EVALUATION

We evaluate our method's capability to (1) create a natural expressive performance given the music score, and to encode an expressive performance into (2) perceptually relevant space that is (3) representative of the corresponding performance. For training, we used an in-house dataset comprising of 380 classical pieces, mostly Late romantic and Baroque. The piano performance comprised of 130 in-house performances and 250 performances obtained from the piano e-competition archive [1]. Some performances were different interpretations of the same piece. Furthermore, corresponding music scores were obtained. To generate a one-to-one mapping between the performance and the score, each performance was aligned to the music score using symbolic music alignment based on [16] with manual alignment correction. Then, the notes in the performance and the score were matched using maximum bipartite matching, using the pitch and the onset timing to determine the edge weights between notes in the score and the performance.

We have used 370 of 380 pieces for training. Of 370 pieces, all but 500 notes were used for training, and remaining notes for the optimization of the hyperparameters (validation). To test the decoder in experiment 1 and 2, we have sampled 10 piano music scores from the Mutopia project [2], and corrected the key signature, meter and downbeat positions. It contained a wide variety of pieces from Baroque to Ragtime. To test the encoder in experiment 3, we have used 10 pieces of 380 pieces that were not used for training.

4.1 Listening test

We evaluated the naturalness of the generated performance using different methods. For each of the ten pieces, we extracted a random 15-second segment, and generated performance using three methods: (1) method **Deadpan** directly played back the SMF data of the music score data that has been exported from MakeMusic's Finale Version 25; (2) method **Finale** used "Human Playback" function of Finale to create a natural performance, serving as a reproducible reference music performance method; and (3) the **proposed** method. Furthermore, we extracted seven 15-second excerpts from human playback in the training dataset, which serves as an **oracle** method. The selected pieces were different from the first three methods, because the human playing data to the first three were not always available. To make a fair comparison, we removed the sustain pedal data from the oracle, because the pedal is contained in none of the other methods.

Next, the MIDI data were synthesized using a high-quality piano synthesizer and presented in a random order to 11 participants. The participants were asked to evaluate the naturalness and expressiveness, on a rating from 1 to 5, 1 being unnatural or unexpressive, 3 being moderately natural or contains some expression, and 5 being the most

[1] www.piano-e-competition.com
[2] www.mutopiaproject.org

Table 1. Mean opinion scores of the performances

	Deadpan	Finale	Proposed	Oracle
Naturalness	3.02	3.08	**3.29**	3.23
Expressiveness	2.75	2.98	3.14	**3.62**

natural or expressive. The participants were between age 25 and late fifties, working on music technology.

The results are shown in Table 1. Kruskal-Wallis H-test was used for test of significance ($p < 0.01$), since Shapiro test showed non-normality. We found that the effects were significant for both expressiveness and naturalness, for all of the method pairs.

This shows that the system is capable of generating a performance that is as natural as human playing, but the expression has a room for improvement. One possible reason is that the system does not make use of music score markings like expression and phrase markings. The audio on our demo webpage suggests nonetheless that the system learns to "improvise" a noticeable evolution of musical expression over a timespan of multiple notes.

4.2 Test on the perception of the interpretation vector

Second, we analyzed the capability of the interpretation sequence to create a *perceptually distinguishable and consistent* space of musical interpretation.

First, a pair of non-identical and non-overlapping segments were sampled from a piece, which we call segments A and B. Second, a vector Δ was sampled from a 5-dimensional, zero-mean and unit-variance Normal distribution. Third, two interpretations are generated for each of A and B, where rendition A_1 and B_1 are generated with interpretation vector $\mu(h_{n-1}) + \Delta \otimes \exp(\gamma(h_{n-1}))$, and A_2 and B_2 with $\mu(h_{n-1}) - \Delta \otimes \exp(\gamma(h_{n-1}))$. Fourth, A_1, A_2, B_1 and B_2 were successively presented to the participants, randomly presenting B_2 before B_1. They were asked to identify whether the relationship of music expression between $A_1 \rightarrow A_2$ is the same as $B_1 \rightarrow B_2$ or $B_1 \leftarrow B_2$. The participants rated the perceived relationship with a confidence on a two-point scale 2 (0 means cannot tell, 2 means strongly confident about the response). We repeated this experiment on 9 other pairs of segments. After the experiment, we changed the signs of the responses, such that negative confidence indicates the wrong guess ($A_1 \rightarrow A_2$ is $B_1 \leftarrow B_2$), and positive indicates the right guess ($A_1 \rightarrow A_2$ is $B_1 \rightarrow B_2$).

The average rating was 0.34, meaning that there is an agreement between the change of interpretation vector and human perception of interpretation. To test the significance, Wilcoxon signed rank test was used under the null hypothesis that the median rating is zero, since Shapiro test showed non-normality. The effect was significant with $p < 0.01$.

This shows that the encoder does capture a perceptually coherent space of music interpretation, and its modification creates a perceptually consistent difference in the interpretation. Qualitatively, we found that by changing the interpretation vector, there are simultaneous and smooth changes not only in the dynamics and the tempo but also nontrivial aspects like breaking of the chords or the articulation of the accompaniment.

Table 2. Pearson's correlation coefficient between the generated performance and the true performance.

	Num. notes	$z_n = 0$	$z_n \sim$ prior
Velocity	10	0.61	0.72
	40	0.37	0.68
Onset deviation	10	0.02	0.07
	40	0.19	0.24
Duration	10	0.77	0.77
	40	0.82	0.82
BPM	10	0.07	0.21
	40	0.00	0.14

4.3 Test on the predictive capability of the encoder

Finally we evaluated the capability to encode a given performance, by priming h and z with an encoded true human performance, and comparing the true human and the generated performances. First, we primed the decoder by feeding to the encoder the first 20 notes to a performance to infer the initial value of z_n. Next, the remaining 10 or 40 notes were decoded using two different decoders: (1) use z_n sampled from the prior that has been primed in the first step, and (2) fix $z_n = 0$. We repeated this experiment on 100 different starting points, and evaluated the Pearson's correlation coefficient between the generated and human performance.

Table 2 shows the correlation coefficients. The result shows that the encoder is capable of encoding information pertinent to music performance. Significant improvements are seen for the velocity and the tempo, showing that the interpretation space is captures these aspects. At the same time, there are still rooms for improving the prediction of the onset deviation and the tempo.

This shows that we can indeed feed a reference performance snippet to the system, and the system would generate a performance in style of the snippet.

5. CONCLUSION

We have presented music performance rendering method that explicitly encodes underlying sources of expression variations. Our method opens door to wider possibilities for the co-creation of music performance between a machine and humans, by enabling an interpretable and adjustable performance rendering and analysis system. We also believe the capability to decouple musical intent and its execution given the intent opens door to a new abstraction layer for performance analysis.

Future work includes the inference of additional outputs like the pedals and note-off velocity, better expressiveness through incorporation of music score markings, disentanglement of the latent space and interfaces for interactive music performance rendering.

6. REFERENCES

[1] Sergio Canazza, Giovanni De Poli, and Antonio Roda. Caro 2.0: An interactive system for expressive music rendering. *Advances in Human-Computer Interaction*, 2015:1–13, 2015.

[2] Carlos Cancino-Chacón and Maarten Grachten. The basis mixer: a computational romantic pianist. In *Late-Breaking Demo, ISMIR*, 2016.

[3] Carlos Cancino-Chacón and Maarten Grachten. A computational study of the role of tonal tension in expressive piano performance. In *Proc. ICMPC*, 2018.

[4] Carlos E. Cancino-Chacón, Maarten Grachten, Werner Goebl, and Gerhard Widmer. Computational models of expressive music performance: A comprehensive and critical review. *Frontiers in Digital Humanities*, 5:25, 2018.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. EMNLP*, pages 1724–1734, 2014.

[6] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *NIPS*, pages 2980–2988, 2015.

[7] Simon Dixon, Werner Goebl, and Gerhard Widmer. The "Air Worm": an interface for real-time manipulation of expressive music performance. In *Proc. ICMC*, 2005.

[8] Sebastian Flossmann, Maarten Grachten, and Gerhard Widmer. Expressive performance rendering: Introducing performance context. *Proc. SMC*, pages 155–160, 2009.

[9] Anders Friberg and Erica Bisesi. Using computational models of music performance to model stylistic variations. *Expressiveness in music performance: Empirical approaches across styles and cultures*, pages 240–259, 2014.

[10] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. CVPR*, pages 4700–4708, 2017.

[11] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. Graph neural network for music score data and modeling expressive piano performance. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proc. ICML*, pages 3060–3070, 2019.

[12] Tae Hun Kim, Satoru Fukayama, Takuya Nishimoto, and Shigeki Sagayama. Polyhymnia: An automatic piano performance system with statistical modeling of polyphonic expression and musical symbol interpretation. In *Proc. NIME*, pages 96–99, 2011.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[14] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proc. ICLR*, 2014.

[15] Akira Maezawa. Deep linear autoregressive model for interpretable prediction of expressive tempo. In *Proc. SMC*, pages 364–371, 2019.

[16] Akira Maezawa and Kazuhiko Yamamoto. MuEns: A multimodal human-machine music ensemble for live concert performance. In *Proc. CHI*, pages 4290–4301, 2017.

[17] Dasaem Jeong Taegyun Kwon Juhan Nam. VirtuosoNet: A hierarchical attention RNN for generating expressive piano performance from music score. In *Workshop on Machine Learning for Creativity and Design, NeurIPS*, pages 1–6, 2018.

[18] Kenta Okumura, Shinji Sako, and Tadashi Kitamura. Laminae: A stochastic modeling-based autonomous performance rendering system that elucidates performer characteristics. In *Proc. ICMC*, 2014.

[19] Caroline Palmer. Music performance. *Annual review of psychology*, 48(1):115–138, 1997.

[20] Ian Simon and Sageev Oore. Performance RNN: Generating music with expressive timing and dynamics. `https://magenta.tensorflow.org/performance-rnn`, 2017.

[21] Sam Van Herwaarden, Maarten Grachten, and W Bas De Haas. Predicting expressive dynamics in piano performances using neural networks. In *Proc. ISMIR*, pages 45–52, 2014.

[22] Gerhard Widmer, Sebastian Flossmann, and Maarten Grachten. YQX plays Chopin. *AI Magazine*, 30(3):35, 2009.