



## Article

# DExter: Learning and Controlling Performance Expression with Diffusion Models

Huan Zhang <sup>1,\*</sup> , Shreyan Chowdhury <sup>2</sup> , Carlos Eduardo Cancino-Chacón <sup>2</sup> , Jinhua Liang <sup>1</sup> , Simon Dixon <sup>1</sup>  and Gerhard Widmer <sup>2</sup> 

<sup>1</sup> Centre for Digital Music, Queen Mary University of London, London E1 4NS, UK

<sup>2</sup> Institute of Computational Perception, Johannes Kepler Universität Linz, Linz 4040, Austria

\* Correspondence: huan.zhang@qmul.ac.uk

**Abstract:** In the pursuit of developing expressive music performance models using artificial intelligence, this paper introduces DExter, a new approach leveraging diffusion probabilistic models to render Western classical piano performances. The main challenge faced in performance rendering tasks is the continuous and sequential modeling of expressive timing and dynamics over time, which is critical for capturing the evolving nuances that characterize live musical performances. In this approach, performance parameters are represented in a continuous expression space, and a diffusion model is trained to predict these continuous parameters while being conditioned on a musical score. Furthermore, DExter also enables the generation of interpretations (expressive variations of a performance) guided by perceptually meaningful features by being jointly conditioned on score and perceptual-feature representations. Consequently, we find that our model is useful for learning expressive performance, generating perceptually steered performances, and transferring performance styles. We assess the model through quantitative and qualitative analyses, focusing on specific performance metrics regarding dimensions like asynchrony and articulation, as well as through listening tests that compare generated performances with different human interpretations. The results show that DExter is able to capture the time-varying correlation of the expressive parameters, and it compares well to existing rendering models in subjectively evaluated ratings. The perceptual-feature-conditioned generation and transferring capabilities of DExter are verified via a proxy model predicting perceptual characteristics of differently steered performances.

**Keywords:** piano performance; expressive rendering; diffusion model; deep learning; music



**Citation:** Zhang, H.; Chowdhury, S.; Cancino-Chacón, C.E.; Liang, J.; Dixon, S.; Widmer, G. DExter: Learning and Controlling Performance Expression with Diffusion Models. *Appl. Sci.* **2024**, *14*, 6543. <https://doi.org/10.3390/app14156543>

Academic Editor: Luigi Portinale

Received: 20 June 2024

Revised: 17 July 2024

Accepted: 23 July 2024

Published: 26 July 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

A trained musician can take a piece of music and interpret it in their own way, molding and varying the emotional expression of the piece by subtly changing performance parameters. Parametric dimensions include timing, dynamics, articulation, and the use of devices like sustain pedals for piano. Studying such expression patterns has long been of keen interest to musicians, educators, and researchers, and it presents a compelling inquiry into exploring whether such intricate expressions can be accurately encapsulated and replicated using computational systems [1]. The accurate replication of human musical expression through machines not only bridges the gap between traditional and technological approaches to music but also opens new avenues for interactive performances [2] and music education systems [3]. Leveraging such technology can enhance musical training, allowing students and professionals alike to experiment with and learn from dynamically generated expression, thus broadening both creative perspectives and educational methods. In this paper, we aim to render such an expressive performance of a piece of music from its score, using a machine learning model. We propose **DExter**, a *diffusion-based expressive performance generator* that predicts expression parameters conditioned on a score. In addition, we investigate whether the rendering process can also be conditioned on desired

high-level performance characteristics in the form of *mid-level perceptual features* [4,5]. In this way, we are permitted to control the general performance character, as well as explore the potential of a style transfer within the varied space of human expressive performances.

Building upon the foundational studies in expressive performance rendering, significant strides have been made in transitioning from traditional sequence-modeling techniques, such as RNNs and LSTMs [6], to the incorporation of variational autoencoders (VAEs) [7]. A persistent challenge in this domain is the accurate and dynamic representation of musical expression over time, particularly in capturing the subtle and complex interplay between different performance parameters that evolve throughout a piece. In this context, we leveraged the diffusion processes to capture the nuanced temporal fluctuations of expressive performance parameters. By employing a novel conditional architecture within the diffusion framework, DExter enhances the transferability of expressive nuances from a source to a target performance, thereby enriching the model's applicative potential in both performance reproduction and real-time interactive systems.

This paper offers three contributions (project demo page with examples: <http://bit.ly/4a1xs1x> (accessed on 22 July 2024); the code is available at <https://github.com/anusfoil/DExter> (accessed on 22 July 2024)): (1) we propose a diffusion-based method for learning and conditioning the expression parameters in Western classical solo piano performance; (2) we conduct a comprehensive quantitative evaluation of the rendered outputs, along with other renderers, in the literature (re-trained to make for a fair comparison), taking into account multiple expressive dimensions such as asynchrony and articulation; and (3) we explore mid-level conditioned generation and style transfer with our model, conducting an experimental study on the conditioning effects.

## 2. Related Work

### 2.1. Expressive Performance Rendering

Expressive performance rendering has long been a challenge for music information retrieval (MIR) research. While the role of machine learning in such a task was recognized early on [1], several rule-based methods have been proposed and investigated over the years [8–10]. Early experiments in deep-learning-based performance rendering [6,11,12] used traditional sequence modeling architectures like RNNs and LSTMs with modifications focusing on the music hierarchy and score features applied as inputs. Recently, transformer-based systems [13,14] have been proposed for controllable rendering, predicting different aspects of performance, such as the shape of expressive attributes [13] and performance direction markings in a score [14]. All the above systems predict descriptors designed to capture expressive aspects of musical performance, typically tokens representing local tempo and timing deviations. However, such tokenized and quantized encodings of performance parameters are not lossless, and they can result in a large vocabulary to train [7].

Regarding the evaluation of performance-rendering systems, there has been growing criticism of the practice of evaluating against a single ground truth and ignoring the variations in interpretations [15], as reconstruction-based error analysis has inherent limitations to fidelity and diversity [15,16]. To mitigate this problem, we evaluated the rendered performances with respect to a multitude of performance-parameter dimensions and against multiple different human interpretations.

### 2.2. Diffusion Models in Music

Diffusion probabilistic models (DPMs) generate data by inverting a Markovian data corruption process. DPMs have demonstrated impressive results, first in the vision domain by generating text-controlled images [17], and then also in the audio domain, with the most promising applications involving the generation of high-fidelity audio samples [18,19] and the synthesis of speech [20] and music [21].

Symbolic music, however, seems to be a more challenging target for DPMs—the challenge is to fit their probabilistic formulation into discrete data distributions.

Mittal et al. [22] train continuous DDPMs (denoising diffusion probabilistic models) on sequences of latent MusicVAE [23] embeddings in order to achieve the generation of monophonic melodies. Plasser et al. [16] build upon the MusicVAE-like token representation and directly apply discrete denoising diffusion probabilistic models (D3PM). Another representation suitable for learning symbolic music [24] under the DPM framework is the piano roll: Cheuk et al. [25] managed to transcribe music by generating a piano roll using an audio spectrogram as a condition. Min et al. [26] also achieved piano roll generation with more diverse control, such as infilling the music context and the high-level guidance of chords and texture.

Our work places music DPMs into a niche spot: while the rendering is applied to symbolic data (discrete notes), DExter predicts continuously varying expressive parameters that are then applied at the note level. The generation of continuous expressive parameters facilitates the fine-grained control of the performance parameters of each note without the reverse diffusion process having to learn a quantized representation space.

### 3. Methodology

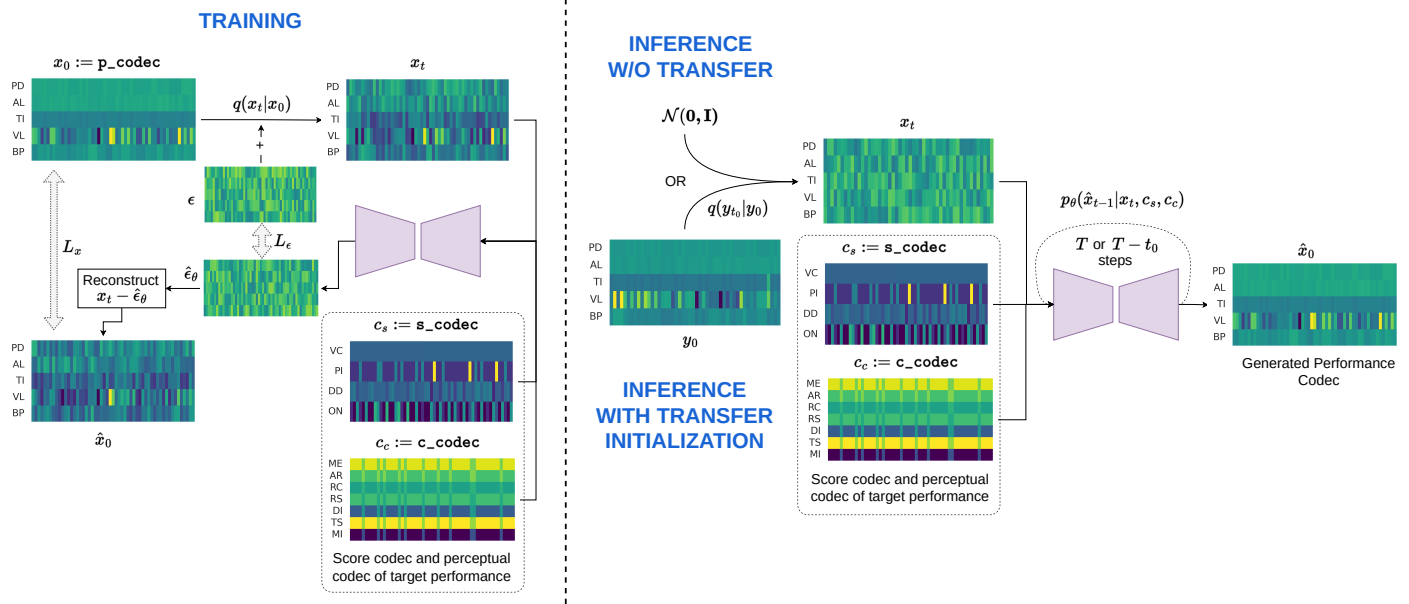
In this methodology section, we begin with our input, codec representations—specifically the score codec (`s_codec`), performance codec (`p_codec`), and perceptual features codec (`c_codec`)—formatted as two-dimensional, spectrogram-like matrices. These codecs, detailed in subsequent subsections, serve as the foundational input and output representations that facilitate the precise handling of musical scores and expressive parameters within a deep learning framework. Following this, we describe our diffusion process, selected for its capability to effectively model the complex, continuous dynamics of musical expression through advanced forward and reverse transformations. Completing the section, we elaborate on the architecture and conditioning strategies of our model, which are centered around a robust 2D UNet architecture. This setup is enhanced by a joint conditioning layer that integrates score and perceptual information, enabling the system to generate nuanced and dynamically expressive music. Each of these components is detailed in dedicated subsections, providing a comprehensive overview of our approach to tackling the challenges of expressive music performance rendering.

#### 3.1. The Codecs

We represent score information (note onset, duration, pitch, and voice), performance parameters (beat period, velocity, timing, articulation ratio, and pedal), and mid-level perceptual features (melodiousness, articulation, rhythmic complexity, rhythmic stability, dissonance, tonal stability, and minoriness) as two-dimensional, spectrogram-like matrices of (mostly real-valued, except for the score codec) numeric values. We call these the **score codec** (`s_codec`), the **performance codec** (`p_codec`), and the **perceptual features codec** (`c_codec`), respectively. The task of our diffusion model is to predict a `p_codec` conditioned on the `s_codec` and `c_codec`. Detailed descriptions of the composition of the codecs are given in Section 4.1.

#### 3.2. Diffusion Framework

We frame the expression-rendering problem as the task of learning a continuous space of performance expression parameters, as shown in Figure 1. Diffusion models [27] consist of two processes: (i) a forward process that transforms each data sample into a standard Gaussian noisy sample, step by step, with a predefined noise schedule; and (ii) a reverse process in which the model learns to denoise pure-noise inputs gradually, generating samples from the learned training data distribution. In effect, our model aims to convert Gaussian noise  $x_t$  into a posterior performance codec,  $\hat{x}_0$ , conditioned on a score codec,  $c_s := s\_codec$ , and a perceptual features codec,  $c_c := c\_codec$ .



**Figure 1.** Training (left) and inference (right) phases of the diffusion framework. Training starts with p\_codec and corrupts the  $x_0$  by injecting noise; the UNet model takes in the corrupted conditions of  $x_t$ ,  $c_s$  and  $c_c$ , to predict the injected noise, which is then used to reconstruct  $\hat{x}_0$ . The loss is calculated for both noise prediction and p\_codec reconstruction. The inference process (right) starts with a random sample from  $\mathcal{N}(0, \mathbf{I})$ ; the model iteratively predicts the noise and reconstructs  $\hat{x}_0$ , conditioned on the same  $c_s$  and  $c_c$ . Alternatively, for transferal, we initialize the process from another performance,  $y_0$ , corrupting it for  $t_0$  steps and denoising for the remaining  $T - t_0$  steps.

The **diffusion forward pass**  $q(x_t|x_0)$  produces a probability distribution of progressively noisier versions of the performance codec, conditioned on the original. With the noise  $\epsilon \in \mathcal{N}(0, \mathbf{I})$  sampled from a standard Gaussian distribution, we blend it with the input sample  $x_0$ , using  $\beta$  as a scaling factor intended to ultimately achieve a zero mean and unit variance of the fully noised result. Specifically, the sampling process applies a linear noise schedule with  $\beta \in [0.0001, 0.2]$ . As we would like to perform multiple steps simultaneously, reparameterization is applied to derive a closed-form equation, given that  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha} = \prod_{s=1}^t \alpha_s$ .

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon \quad (1)$$

During **training**, model  $f_\theta(x_t, t, c_s, c_c)$  learns to predict the injected noise  $\hat{\epsilon}_\theta$ , given a random timestep,  $t$ , and its noised codec version,  $x_t$ , calculated in the forward pass.  $t$  is sampled from  $[1, T]$ ; we used  $T = 1000$  in our experiments. Then, we use the predicted noise  $\hat{\epsilon}_\theta$  to reconstruct the predicted initial codec  $\hat{x}_0$  by inverting Equation (1). The parameter  $T$ , or the time interval limit, significantly influences the granularity and smoothness of the denoising process. A higher  $T$  allows for a more gradual denoising process, potentially capturing more subtle nuances in the data as it iteratively approximates the clean state.

The training objective combines noise estimation and codec reconstruction, as shown in Equation (2). Although theoretically, noise prediction alone could suffice to train the model, the empirical results demonstrate that adding a constraint to the reconstructed codec significantly improves performance. We use a weighting factor,  $h = 0.2$ , derived empirically, to balance the emphasis between accurate noise prediction and fidelity in codec reconstruction, optimizing both the precision of the generated output and the stability of the learning process.

$$L(\theta) = \|\epsilon - \hat{\epsilon}_\theta\|^2 + h \|x_0 - \hat{x}_0\|^2 \quad (2)$$

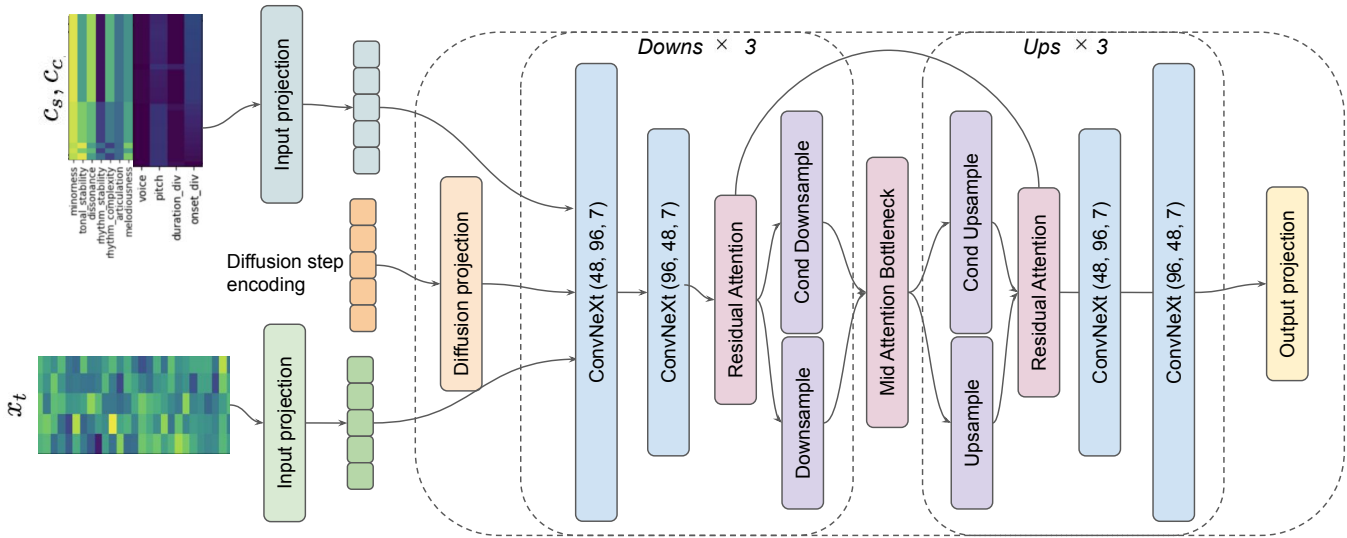
During **inference**, we start from a Gaussian noise distribution,  $p(x_t) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ , and iteratively generate the codec posterior through  $p_\theta(\hat{x}_{t-1}|x_t, c_s, c_c) = \mathcal{N}(\mu_{\theta,t}(x_t, t, c_s, c_c), \sigma_t^2 \mathbf{I})$  until  $\hat{x}_0$  is reached. As the model  $f_\theta$  estimates noise  $\hat{\epsilon}_\theta$ , we use it to construct the model mean  $\mu_{\theta,t}$ , and the posterior variance  $\sigma_t^2$  is predetermined according to the noise schedule. The full construction of model mean and posterior variance is given in Equations (3) and (4):

$$\mu_{\theta,t}(x_t, t, c_s, c_c) = \sqrt{\frac{1}{\alpha_t}}(x_t - \frac{\hat{\epsilon}_\theta(1 - \alpha_t)}{\sqrt{1 - \bar{\alpha}_t}}) \quad (3)$$

$$\sigma_t^2 = (1 - \alpha_t) \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \quad (4)$$

### 3.3. Architecture and Conditioning

We employed a 2D UNet as the backbone of our architecture; the detailed layer and insertion structure can be found in Figure 2. The conditioning on score information and perceptual feature information is enforced by a joint conditioning layer that projects the score dimensions and perceptual dimensions (5 and 7, respectively—see the definition of codecs below) onto 512 dimensions. The diffusion timestep  $t$  is encoded via sinusoidal position embeddings. The input codec and the conditioning codecs are downsampled and upsampled through ResNet blocks and 2D convolutions. Attention layers are interleaved at bottlenecks.



**Figure 2.** Diagram of the UNet conditioning module in the network.

Before narrowing down on the above-described architecture, we experimented with a DiffWave-based architecture [18] that uses a series of 12 residual layers of 1D convolution. For conditioning, we also experimented with FiLM [28,29], which yields comparable results to the UNet model. We found that our present architecture provides the best trade-off between model simplicity and performance.

**Classifier-free guidance (CFG)** [30] is widely used for conditioning diffusion models to achieve controllable generation, and we also adopted it. During training, a dropout layer is applied to the conditions  $c_s$  and  $c_c$  to randomly mask out the conditions with probability  $p$  in order to simultaneously train the conditional model,  $f_\theta(x_t, t, c_s, c_c)$ , and the unconditional model,  $f_\theta(x_t, t)$ . Based on preliminary experiments, we chose a fixed dropout probability of  $p = 0.1$  for our training to optimally balance learning from conditioned data



while maintaining the ability to generate coherently without specific conditioning cues, enhancing model generalization. In inference, a weight parameter,  $w$ , is applied as the guidance scale to a combined prediction.

$$\hat{e} = w\hat{e}(x_t, t) + (1 - w)\hat{e}(x_t, t, c_s, c_c) \quad (5)$$

## 4. Data, Representation, and Processing

### 4.1. Input and Target Encodings

The **performance codec** (p\_codec)—our prediction target—was originally proposed in the expressive rendering framework *Basis Mixer* [11], in which four expressive parameters are computed for each note,  $n$ , appearing in the score. These parameters of the p\_codec encoding the expression controls modify the properties of notes in a MIDI piano performance, thus changing the speed and loudness of the performance with time. Combined with score information, the full expressive performance can be reconstructed in a lossless fashion. We expanded the original Performance Codec v.1.0 [2,11] by defining an additional parameter for sustain-pedal control. The resulting five (note-wise) performance parameters are as follows:

- **Beat period:** The ratio of the inter-onset intervals (IOIs) between two consecutive notes of the performance and the score. This parameter is computed for each onset,  $o_k$ , instead of each note,  $n_i$ . It is defined as follows:  

$$x_{bp}(o_k) = \frac{\text{IOI}_{\text{perf}}(o_k)}{\text{IOI}_{\text{score}}(o_k)} = \frac{\delta_{k+1}^{\text{perf}} - \delta_k^{\text{perf}}}{o_{k+1} - o_k},$$
where  $\delta_k^{\text{perf}}$  is the actual performed onset time, in seconds, corresponding to score onset  $o_k$  in beats, calculated as the average onset time of all notes played at score onset position  $o_k$ .
- **Velocity:**  $x_{\text{vel}}(n_i) = \frac{\text{vel}(n_i)}{127}$ , where  $\text{vel}$  is the MIDI velocity of a played note.
- **Timing:**  $x_{\text{tim}}(n_i) = \Delta_t(n_i) = \delta^{\text{perf}}(n_i) - \text{onset}(n_i)$ , the average onset time of all notes played at the score onset position (used in beat period), minus the performance onset time of  $n_i$ . The beat period is taken as the ‘tempo grid’ notion [31]; timing would then refer to the micro-deviation of each note relative to the grid.
- **Articulation ratio:**  $x_{\text{art}}(n_i) = \frac{\text{dur}^{\text{perf}}(n_i)}{\text{dur}(n_i) \cdot x_{bp}(n_i)}$ , which measures the fraction of the expected note duration that is actually played.
- **Pedal:**  $x_{\text{ped}}(n_i) = \frac{\text{ped}(n_i)}{127}$ , where  $\text{ped}(n_i)$  is the discrete MIDI pedal value at the onset of  $n_i$ . Note that pedal encoding is not lossless since changes in the value between note onsets will not be captured.

The p\_codec is fully invertible in that the full event information from the MIDI file [Pitch, Onset, Duration, and Velocity] can be reconstructed, given the p\_codec and score s\_codec.

The **score codec** (s\_codec) represents the musical score, and it is derived from the note array from the *partitura* package [32]. Aligned with the p\_codec at the note level, it contains four score parameters for each note: (notated) Onset, Duration, Pitch, and Voice, resulting in a 2D matrix of dimension  $4 \times n$ , where  $n$  is the number of notes. The score is indispensable for performance conditioning, as it defines the musical content of the piece.

The **perceptual features codec** (c\_codec), which we use as a steering input for performance generation, is a representation of the so-called *mid-level perceptual features* [4], namely *melodiousness*, *articulation*, *rhythm complexity*, *rhythm stability*, *dissonance*, *tonal stability*, and ‘*minorness*’ (or *mode*). They describe musical qualities that most listeners can easily perceive. Taking a cue from previous research [5,33] showing that these features effectively represent musical factors underlying a wide range of emotions and capture variations in the expressive character between different performances of a piece [34], we incorporated these as the performance-steering inputs. In our scenario, these features are calculated via a previously trained specialized model [5] over the recorded audio performance data of the Vienna4×22, (n)ASAP, and ATEPP datasets (see Section 4.4). The values are calculated from successive overlapping 15-s windows with a hop size of 5 s. Each computed window is then aligned with the score note array to be broadcast into c\_codec, a 2D matrix of dimension  $7 \times n$ .

#### 4.2. Processing

Given that there could be slight variations in each performance (missing and extra notes relative to the score), we performed padding based on the score note array so that each pair of performances was perfectly aligned. To accommodate pieces of different lengths, we trained our network on segments of  $N$  notes in which shorter segments were padded. In our experiment, we used  $N = 200$  (which corresponds to about 10 to 20 s of music, depending on the tempo and note density).

#### 4.3. Mixup Augmentation

Mixup [35] is a data augmentation scheme that regularizes a network to favor simple linear behavior between training examples. To strengthen our model's ability to model different interpretations, we fused p\_codec pairs  $x_1$  and  $x_2$  (codecs representing two different performances of the same piece) and their corresponding c\_codec using Equation (6), where  $\lambda$  is a scaling factor with a variance of  $[0, 1]$ .

$$x_{1,2} = \lambda x_1 + (1 - \lambda)x_2 \quad (6)$$

After the mixup augmentation, our dataset consisted of 170k segments; the interpolated data were only used in training.

#### 4.4. Datasets and Training Setup

We used three datasets of expressive performances (from the Western classical music solo piano repertoire): Vienna4×22 [36], (n)ASAP [37], and ATEPP [38]. Each dataset includes audio, performance MIDI, a score in MusicXML format, and their alignment. Information and a comparison of these sets can be found in Table 1. The training was based on ATEPP and 80% of the (n)ASAP and Vienna4×22 data, while the testing set (used in all subsequent experiments in Section 5) contained the remaining 20% of (n)ASAP and Vienna4×22 data. The latter two datasets were recorded on computer-controlled grand pianos and are, thus, more accurate and precise than the ATEPP data, which were obtained through curated audio transcription. While these datasets provide comprehensive coverage of certain composers and styles within the Western classical tradition, further validation would be beneficial to determine their applicability to a broader range of composers and musical styles not represented in the current dataset collection.

For the training of the network, as mentioned in Section 3, we used the Adam optimizer with a learning rate of  $5 \times 10^{-5}$ , and we employed early stopping with a patience of 50 epochs.

**Table 1.** Overview of datasets used in experiments.

Dataset	Pieces	Performances	Duration	MIDI	Repertoire
Vienna4×22 [36]	4	88	2 h 18 min	recorded	Excerpts from four pieces by F. Chopin (Op. 10 No. 3 and Op. 38), W. A. Mozart (KV331, first mov.), and F. Schubert (D. 783 No. 15)
(n)ASAP [37]	235	1067	94 h 30 min	Recorded	Common Practice Period solo piano works by 15 composers
ATEPP [38]	1580	11,677	1000 h	Transcribed	Solo piano works by 25 composers, ranging from the Baroque to Modern eras

### 5. Evaluation

In this section, we present a quantitative evaluation of generated performances with-out and with steering, followed by an evaluation of the performance transfer and an investigation into the effect of varying the conditioning weight. Finally, we also describe our qualitative study, employing a listening test and human participants, and we present the results.

### 5.1. Quantitative Evaluation

In this subsection, we evaluate our generated samples' expressiveness by comparing core expression attributes with ground-truth performances. This experiment was conducted using the aforementioned testing set, with the condition of `s_codec` and audio-performance-inferred `c_codec` as described in Section 4.1. With respect to the critique of reconstruction-based evaluation [15], we compared the results with various interpretations of ground truth (the testing set consisted of about 5.3 human performances for each piece, on average).

#### 5.1.1. Assessed Attributes

The expression attributes we assessed were derived from the tempo and velocity curves (joint-onset level), joint-onset asynchrony, articulation, dynamics, and pedaling. While a detailed documentation of the selected attributes can be found on the project page, we provide a summary below.

- Tempo curve: Onset-level tempo (inverse of local inter-beat intervals), with values averaged across notes at the joint onset (**tem\_curve**).
- Velocity curve: Onset-level velocity, with values averaged across notes at the joint onset (**vel\_curve**).
- Asynchrony: The absolute difference in seconds between the earliest and latest notes in a joint onset (**asy.delta**). We also measured the pitch correlation (**asy.p\_cor**) between the pitch and micro-timing within the joint onset, inspired by the *melody lead* phenomenon [39].
- Articulation: The key overlap ratio (**art.kor**) [40], measured at each note transition; the overlap time (or gap time if *staccato*), divided by the IOI between the two notes.
- Dynamics: Comparing the performed velocity and score marking (*f*, *p*, etc.) and measuring their agreement (**dyn.agr**) and consistency (**dyn.con**), as proposed by Kosta et al. [41]. We also propose ramp correlation (**dyn.r\_cor**) for changing markings (hairpins) since Kosta et al. [41] only worked with constant markings. Ramp correlation computes the amount of agreement between the performed velocities with respect to their *cresc.* or *decresc.* ramps, if the markings exist.
- Pedals: We measured the sustain pedal value at the note onset (**ped.onval**). Actually, a sustain-pedal change is a continuous stream of values, and changes in pedal position often happen between note attacks; however, sampling at the note onsets simplifies the computation and allows for a consistent assessment across models.

#### 5.1.2. Metrics

For each expression dimension, we measured three metrics between the generated performance and the ground-truth space.

**Standard-deviation multiple:** This metric computes the deviation of an attribute of the rendered output from the mean of multiple human performances on a beat-level basis. Different from absolute deviation, this measure incorporates the flexibility of interpretations: when human interpretations already contain large differences, a larger discrepancy can be tolerated. But if human players tend to agree on the interpretation, we would expect the rendered output to fit more closely to ground-truth values. Additionally, we retained the sign (direction) of deviation so that negative values indicate deviations in the direction of slower-tempo or softer dynamics, for example.

**KL divergence** takes all the note-level or beat-level attributes, and it compares their divergence with the ground-truth attributes as an overall distribution (note that the ground-truth attribute distributions are aggregated from multiple interpretations). The KL divergence is calculated by estimating the two distributions using Monte Carlo sampling ( $N = 300$ ) and computing the relative entropy between them.



**Pearson's correlation** is measured between the feature sequences of generated and ground-truth performances. In contrast to the previous two metrics, this metric captures the time-varying similarity between the performance attributes.

### 5.1.3. Results

In Table 2, we compare our model with the samples from two existing performance-rendering systems: BasisMixer [11] (BM) (we applied the Basis Mixer model with LSTM architecture trained on the ASAP corpus) and VirtuosoNet [12] (VN) (the applied model is the sign with a default tempo and composer settings agreed on by the author). The results were rendered using the same testing set as ours and shown as means and standard deviations.

**Table 2.** Quantitative expression metrics in the categories of articulation, asynchrony, dynamics, pedaling, plus global tempo and velocity curves. The columns represent different models, and the rows are divided into blocks according to the three different evaluation metrics, with each block detailing the outcomes for all features. Note that each generated performance was compared with multiple human ground-truth interpretations.  $\rightarrow 0$  means a value close to 0 is better.  $\downarrow$  indicates a lower value is preferred while  $\uparrow$  is otherwise. The numbers in bold are the best performance on the metric.

	Basis Mixer [11]	VirtuosoNet [12]	DExter (Ours)
<i>Deviation Multiple (<math>\rightarrow 0</math>)</i>			
articulation.key_overlap_ratio	<b>0.62 <math>\pm</math> 3.15</b>	1.92 $\pm$ 2.72	2.24 $\pm$ 2.76
asynchrony.pitch_correlation	−1.19 $\pm$ 1.41	−1.67 $\pm$ 1.25	<b>−1.17 <math>\pm</math> 1.66</b>
asynchrony.delta	<b>4.10 <math>\pm</math> 1.78</b>	4.38 $\pm$ 1.63	4.43 $\pm$ 2.21
dynamics.agreement	−0.07 $\pm$ 1.33	−0.40 $\pm$ 1.30	<b>−0.002 <math>\pm</math> 1.05</b>
dynamics.consistency	<b>−0.67 <math>\pm</math> 1.64</b>	−1.07 $\pm$ 1.56	0.73 $\pm$ 2.21
dynamics.ramp_correlation	−0.36 $\pm$ 2.54	0.65 $\pm$ 1.96	<b>−0.28 <math>\pm</math> 2.44</b>
pedal.onset_value	-	−1.16 $\pm$ 1.68	−1.39 $\pm$ 2.13
<b>tempo_curve</b>	<b>−0.10 <math>\pm</math> 2.44</b>	0.52 $\pm$ 2.48	0.72 $\pm$ 2.65
<b>velocity_curve</b>	−0.67 $\pm$ 1.48	<b>0.15 <math>\pm</math> 1.02</b>	1.48 $\pm$ 2.12
<i>KL Divergence (<math>\downarrow</math>)</i>			
articulation.key_overlap_ratio	<b>0.92 <math>\pm</math> 2.15</b>	1.66 $\pm$ 6.89	1.64 $\pm$ 3.63
asynchrony.pitch_correlation	0.14 $\pm$ 0.278	<b>0.13 <math>\pm</math> 1.25</b>	0.20 $\pm$ 0.33
asynchrony.delta	4.04 $\pm$ 5.15	4.83 $\pm$ 9.58	<b>1.29 <math>\pm</math> 3.16</b>
dynamics.agreement	0.10 $\pm$ 0.04	0.09 $\pm$ 0.04	<b>0.06 <math>\pm</math> 0.04</b>
dynamics.consistency	0.12 $\pm$ 0.23	<b>0.06 <math>\pm</math> 0.07</b>	0.28 $\pm$ 0.49
dynamics.ramp_correlation	1.54 $\pm$ 5.43	<b>0.35 <math>\pm</math> 1.01</b>	0.42 $\pm$ 1.13
pedal.onset_value	-	0.34 $\pm$ 1.45	0.33 $\pm$ 0.36
<b>tempo_curve</b>	0.98 $\pm$ 2.55	<b>0.65 <math>\pm</math> 1.86</b>	1.26 $\pm$ 5.66
<b>velocity_curve</b>	0.16 $\pm$ 0.21	<b>0.10 <math>\pm</math> 0.06</b>	0.71 $\pm$ 1.37
<i>Pearson's Correlation (<math>\uparrow</math>)</i>			
articulation.key_overlap_ratio	−0.01 $\pm$ 0.13	0.05 $\pm$ 0.16	<b>0.11 <math>\pm</math> 0.17</b>
asynchrony.pitch_correlation	0.33 $\pm$ 0.25	0.55 $\pm$ 0.17	<b>0.57 <math>\pm</math> 0.25</b>
asynchrony.delta	0.17 $\pm$ 0.22	<b>0.29 <math>\pm</math> 0.19</b>	0.28 $\pm$ 0.21
dynamics.agreement	0.02 $\pm$ 0.87	0.04 $\pm$ 0.84	<b>0.11 <math>\pm</math> 0.79</b>
dynamics.consistency	0.92 $\pm$ 0.17	0.91 $\pm$ 0.13	<b>0.92 <math>\pm</math> 0.15</b>
dynamics.ramp_correlation	0.04 $\pm$ 0.76	0.12 $\pm$ 0.80	<b>0.14 <math>\pm</math> 0.73</b>
pedal.onset_value	-	0.01 $\pm$ 0.13	0.02 $\pm$ 0.14
<b>tempo_curve</b>	0.02 $\pm$ 0.13	0.09 $\pm$ 0.19	<b>0.19 <math>\pm</math> 0.17</b>
<b>velocity_curve</b>	0.08 $\pm$ 0.23	0.21 $\pm$ 0.27	<b>0.27 <math>\pm</math> 0.23</b>

Overall, DExter showed commendable results, particularly in correlation across almost all performance dimensions, especially in capturing the global curve of tempo and velocity. This latter effect (learning the overall musical shape) could be attributed to the diffusion model predicting the time-varying codec in one pass, in contrast to autoregressive approaches. However, it is evident that DExter has room for improvement in terms of deviation and divergence: DExter's outputs demonstrate more volatile changes in parameters that are less smooth than other renderings.

Meanwhile, each model exhibited distinct strengths across various performance dimensions. BM's outputs had an articulation closer to human ground truths, and this can be attributed to BM being more conservative in its use of expressive devices, using smaller deviations from the mechanical reproduction of a score. VN also excels in modeling the dynamics in agreement with score markings. It is also notable that both models with sustain pedal prediction did a poor job in mimicking human pedaling techniques, with almost no time-wise correlation, and on average, they were one standard deviation away from the ground truth. Another area where all models struggled was asynchrony time (asy.delta:  $\sim$ four deviations away from the human benchmark), highlighting the need to refine the micro-timing aspect in performance-rendering models.

While different models may exhibit varied performance levels with specific types of compositions, DExter is designed to handle a broad spectrum of inputs. The differences in performance metrics reflect the diverse interpretative approaches that each model embodies, illustrating that the evaluation of music-performance-rendering models is not solely about surpassing benchmark metrics but also about enriching the diversity and depth of musical expression.

## 5.2. Expressive Steering with Perceptual Features

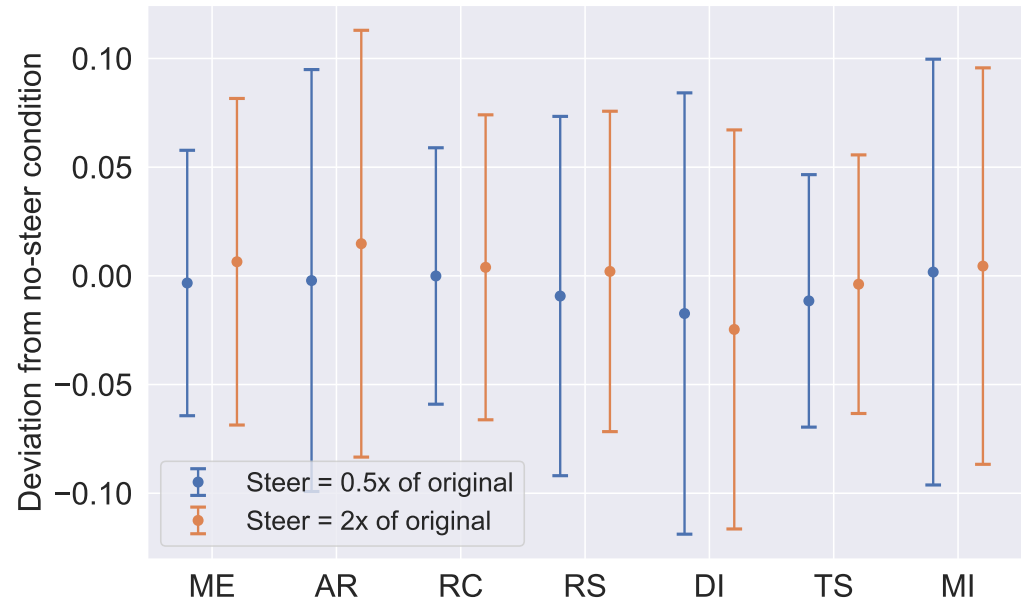
Our framework of conditioned performance generation allowed us to explore the conditioning of the performance generation using additional features. As described in Section 4.1, we used mid-level perceptual features (encoded as  $c\_codec$ ) as steering inputs to guide the expressive character of the generated performance.

To gauge DExter's sensitivity to changes in these features, we used the perceptual feature recognition model of [5] as a proxy for human perception. However, as that model had originally been trained on an audio input, we wanted to eliminate the effect of acoustic artifacts introduced by rendering MIDI to audio; we decided to fit a MIDI-to-perceptual-features model to serve as the proxy instead. Details on this proxy model are given in Appendix A.

Steering performance generation is achieved by manipulating the individual dimensions of perceptual features, aiming to induce measurable corresponding effects in the resulting outputs. For each test sample and across all seven perceptual attributes, we first generated performances using the unmodified target  $c_c$ . These target perceptual-feature values could be randomly initialized in practice or derived from an actual performance. We took the feature values derived from actual performances and modified the values to steer the generation, in particular expressive directions, thus generating 'alternate' performances of the original performance. We either halved one feature,  $\frac{1}{2}c_c$ , or doubled one feature,  $2c_c$ , at a time.

Figure 3 displays the proxy model's predictions of the generated outputs. We observed that, for the first four features *melodiousness*, *articulation*, *rhythm complexity*, and *rhythm stability*, the adjustments applied to the input conditions manifested as anticipated directional changes (the  $c_{c,double}$  group led the  $c_{c,half}$  group at 12.2% in terms of their absolute values), providing evidence of the model's responsive behavior to the controlled feature alterations.

The other three dimensions—notably, *dissonance*—exhibited less consistent patterns in alignment with the input modifications. That seems reasonable, as harmonic and tonality-related properties are more functions of a piece itself, rather than any specific interpretation of it.



**Figure 3.** Steering the expressive characteristics of generated performances using mid-level features ( $c\_codec$ ; see Section 4.1) for conditions. For each piece, a performance is generated with the  $c\_codec$  derived from an actual performance of the piece, and two further performances are generated with one of the mid-level features doubled ( $2\times$ ) or halved ( $0.5\times$ ). The average difference between the halved and unmodified conditions, and between the doubled and unmodified conditions, are plotted here.

### 5.3. Transferring from a Source Performance

As suggested by Liu et al. [42] and Zhang et al. [43], a style transfer can be achieved in a diffusion framework by using, as a starting point for generation, a shallowly noised version of the source information. Given the large amount of music overlap in our datasets, we could test this by forming data pairs that consisted of two interpretations of the same piece to be used as the source and target  $p\_codec$  in this experiment.

Given a source performance codec,  $x_{src}$ , we calculated its noisy version,  $x_{t_0}$ , with a predefined time step,  $t_0 \leq T$ , according to the forward process shown in Equation (1). By using  $x_{t_0}$  as the starting point for the reverse process of a pretrained model, we enabled the manipulation of performance  $x_{src}$  with a target mid-level condition and a shared score condition,  $c_{(s,tgt)}$ , in a shallow reverse process,  $p_{\theta}(\hat{x}_{0:t_0} | x_{t_0}, c_{(s,tgt)})$ , as illustrated in Figure 1 (top right). With the transfer experiments, we attempted to understand the following two questions:

1. Does transferring help with the final generation quality compared with rendering from scratch?

In the transfer experiment, we combined pairs of ground-truth performances of the same piece segment,  $x_{src}, x_{tgt}$ , where  $s_{src} = s_{tgt}$ . The same testing set as that used in the previous sections was used, and the source performance was randomly taken from the ground truth. We experimented with different transfer steps of  $t_0 \in \{T, \frac{3T}{4}, \frac{T}{2}, \frac{T}{4}\}$ , and we reported the global metrics of the tempo curve and velocity curve with their deviation and correlation.

What we observed in Table 3 was that transferring from a source performance slightly helped with initialization. Specifically, employing a denoiser for three-quarters of the diffusion steps—ideally preserving around one-quarter of the source’s characteristics—yielded the highest-quality outcomes. However, the transfer quality did not steadily

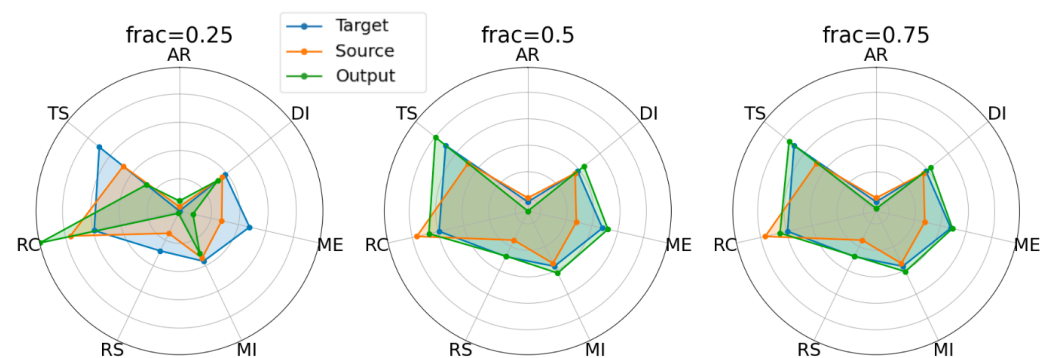
improve with the retained information from the source: the  $\frac{T}{4}$ -step transfer resulted in ambiguous outputs that did not align well with the given score.

**Table 3.** Deviation and correlation of test set relative to ground-truth space (same analysis as in Section 5.1) of tempo and velocity curves. ↓ indicates a lower value is preferred while ↑ is otherwise. The numbers in bold are the best performance on the metric.

$t$	<i>dev: tem</i> (↓)	<i>dev: vel</i> (↓)	<i>cor: tem</i> (↑)	<i>cor: vel</i> (↑)
$T$	$0.72 \pm 2.65$	$1.48 \pm 2.13$	$0.19 \pm 0.17$	$0.27 \pm 0.23$
$\frac{3T}{4}$	<b><math>0.68 \pm 2.55</math></b>	$1.40 \pm 2.16$	<b><math>0.19 \pm 0.16</math></b>	<b><math>0.28 \pm 0.21</math></b>
$\frac{T}{2}$	$0.74 \pm 2.49$	<b><math>1.33 \pm 2.10</math></b>	$0.15 \pm 0.17$	$0.21 \pm 0.22$
$\frac{T}{4}$	$0.87 \pm 2.69$	$1.50 \pm 2.11$	$0.11 \pm 0.16$	$0.18 \pm 0.21$

## 2. Does a transferred rendering sound ‘closer’ to the source or the target?

We wished, similar to Section 5.2, to measure the transfer proximity using the predicted perceptual features. The radar plots in Figure 4 show the seven perceptual feature dimensions predicted via the proxy, illustrating the perceptual distance between the source, target, and generated performance for three different transfer gradations,  $\frac{T}{4}$ ,  $\frac{T}{2}$ ,  $\frac{3T}{4}$ . At  $\frac{T}{4}$  steps, the predicted performance deviated from both the source and target, which fit our previous observation that insufficient denoising steps result in ambiguous outputs. As the transfer step increases, there is a discernible shift in the predicted output towards the target profile across most perceptual dimensions.



**Figure 4.** Seven dimensions of perceptual features (AR: articulation; RC: rhythm complexity; RS: rhythm stability; TS: tonal stability; DI: dissonance; MI: minoriness; and ME: melodiousness) predicted using the proxy for the output, source, and target, averaged across the testing set. The three plots correspond to the transfer steps of  $0.25T$ ,  $0.5T$ ,  $0.75T$ .

## 5.4. Ablation: Effect of Varying Conditioning Weights

In this ablation experiment, we looked at the effect of the conditioning weight  $w$  on the generated results. As described in Section 3.3 and Equation (5), the scale of classifier-free guidance,  $w$ , is the ratio that combines the prediction with and without (masked by 0) the  $c_s$  and  $c_c$  conditioning. While the conditional and unconditional models were jointly trained in the training phase, the weighting parameter  $w$  was only introduced in the sampling phase, and the optimal  $w$  was not trivial to find. In Table 4, conditioning weights  $w = 0.5, 1.2, 2, 3$  are compared, while the other settings are the same as in Section 5.1. The experimental results were best for  $w = 1.2$ . The experiments from the ablations justified that  $w = 1.2$  is around the best weighting that generates musically sound output. Interestingly, with a greater scale of classifier guidance, the generative results exhibit larger fluctuations in expressive parameters and less stability.

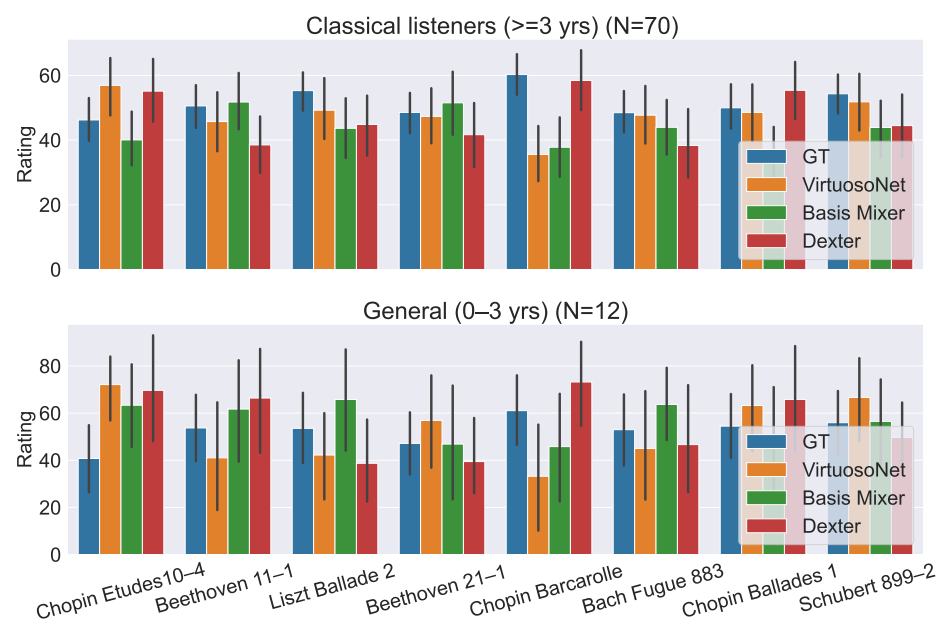
**Table 4.** Deviation and correlation of test set relative to ground-truth space (same analysis as in Section 5.1) of tempo and velocity curves.  $\rightarrow 0$  means a value close to 0 is better.  $\downarrow$  indicates a lower value is preferred while  $\uparrow$  is otherwise. The numbers in bold are the best performance on the metric.

$w$	<i>dev: tem</i> ( $\downarrow$ )	<i>dev: vel</i> ( $\rightarrow 0$ )	<i>cor: tem</i> ( $\uparrow$ )	<i>cor: vel</i> ( $\uparrow$ )
0.5	$1.11 \pm 2.46$	$-2.47 \pm 1.10$	$0.11 \pm 0.16$	$0.02 \pm 0.22$
1.2	<b><math>0.72 \pm 2.65</math></b>	<b><math>1.48 \pm 2.12</math></b>	<b><math>0.19 \pm 0.16</math></b>	<b><math>0.28 \pm 0.21</math></b>
2	$1.33 \pm 2.37$	$3.02 \pm 1.53$	$0.04 \pm 0.15$	$0.13 \pm 0.24$
3	$1.86 \pm 1.81$	$4.63 \pm 1.15$	$0.04 \pm 0.14$	$0.10 \pm 0.23$

### 5.5. Qualitative Study

We evaluated the naturalness and expressiveness of the rendered performances through a listening test. For samples from eight selected pieces, we compared the following: (1) two human performances with relatively distant interpretations; (2) renderings made with Basis Mixer and VirtuosoNet, as described in Section 5.1; and (3) a rendering from the proposed model, DExter. The performances (including the ground truths) were rendered to audio using a Yamaha Disklavier, which produced similar pedal/articulation-related artifacts in both the human and machine performances. Eighty-two participants listened to the performances and evaluated them on a 100-point Likert scale, rating the overall naturalness and expression of the output as one score. The performances used for the test can be found on the demo page.

The results of the listening test, shown in Figure 5, provide a nuanced view of the performance-rendering capabilities of the models in comparison to the ground truth. In terms of the mean rating, DExter demonstrated better performances of the pieces by Chopin, even sometimes comparable to the ground truth (*Barcarolle*). However, they were outperformed via Basis Mixer or VirtuosoNet in the case of older compositions (*Bach's fugue* and *Beethoven's sonatas*). Overall, in terms of the mean rating scores, there was still a gap between the generative outputs and GT (51.81), while DExter (48.54) slightly outperformed VirtuosoNet (48.31) and Basis Mixer (46.33). It was also surprising to observe that GT did not always secure the highest ratings. In the case of Chopin's *étude*, at least, it might be explained by the fact that human pianists suffer from physical limitations in technically demanding passages, while the generative models do not.



**Figure 5.** Mean ratings for the eight pieces evaluated in the listening test for each model and human (ground-truth, GT) performances.



Besides the numerical ratings, we also asked three musically trained participants for explicit feedback on their ratings. In addition to some positive comments, we also received quite specific and useful negative feedback, such as (specifically referring to polyphonic music, such as that of Bach) *no clear voicing among the lines* and *poor balance between the hands*. Given currently dominating ‘flattened’ representations, such as our p\_codec or tokens in transformer models, learning the vertical structure of music remains a challenge for rendering models [24].

## 6. Conclusions and Future Work

In concluding this study, we observe that we have introduced DExter, a novel diffusion-based model adept at learning and controlling performance expression in solo piano music. This model not only matches the quality of existing systems in terms of expressive characteristics but also extends the capability of style transfer and nuanced expression conditioning using perceptual features. However, the current implementation of DExter reveals several limitations that must be addressed. The inference speed, for instance, requires 40 s to process a 95-s piece, which may impede real-time application viability. We plan to explore acceleration techniques such as denoising diffusion implicit models (DDIMs) or embedding the process in a latent space to enhance computational efficiency.

Further, while DExter effectively handles the styled performance transfer and perceptual conditioning, its reliance on highly specific codec representations could limit its generalizability across different musical genres or instrumental configurations. Future iterations could benefit from integrating a broader array of conditioning inputs, such as textual descriptions or broader acoustic features, which might enable the model to adapt more flexibly to varied musical contexts and styles. These adaptations could pave the way for DExter’s application beyond solo piano music, potentially in ensemble settings or for other instruments, enriching interactive music systems and supporting creative musical compositions.

Such expansions could not only address the limitations noted but also broaden the impact of our research within the music technology field, offering new tools and methodologies for researchers, composers, and performers alike.

**Author Contributions:** Methodology, H.Z. and S.C.; Software, H.Z. and J.L.; Validation, S.C.; Investigation, H.Z. and J.L.; Resources, C.E.C.-C.; Data curation, H.Z.; Writing—original draft, H.Z. and S.C.; Writing—review & editing, S.C., C.E.C.-C., S.D. and G.W.; Supervision, C.E.C.-C., S.D. and G.W.; Funding acquisition, S.D. and G.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, funded by UK Research and Innovation [grant number EP/S022694/1] and also by the European Research Council (ERC) under the EU’s Horizon 2020 research and innovation program, grant agreement No. 101019375 (*Whither Music?*).

**Institutional Review Board Statement:** The subjective listening test conducted in this study was approved by the Devolved School Research Ethics Committee (DSREC) of Queen Mary University of London (protocol code QMERC20.565.DSEEC24.001, approved on 16 January 2024).

**Informed Consent Statement:** Informed consent was obtained from all subjects involved in the study.

**Data Availability Statement:** The data presented in this study are openly available: ATEPP: <https://github.com/tangjibetsy/ATEPP> (accessed on 22 July 2024); ASAP: <https://github.com/CPJKU/asap-dataset> (accessed on 22 July 2024); Vienna4×22: <https://github.com/CPJKU/vienna4x22> (accessed on 22 July 2024).

**Acknowledgments:** We would like to thank Daesam Jeong for help with appropriate Parametrisation of VirtuosoNet, as well as Maximillian Hofmann for contributing Basis Mixer trained with LSTM. We’d like to thank musicians Yan Zhou, Ziyu Deng, Guanhong Song and Yuxuan Hu for pilot listening and providing feedback on the renderings.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Appendix A. MIDI-to-Perceptual-Features Model

The MIDI-to-perceptual-features model was used as a proxy for human mid-level perception in the experiments described in Sections 5.2 and 5.3. It takes in a rendered MIDI and outputs seven-dimensional perceptual features for each window of 15 s. The specifications are as follows.

- **Data:** The data used to train this oracle were ASAP performance MIDI, along with the audio-perceptual features computed and predicted from ASAP performance audio via the mid-level feature recognition model of [34] for 15-s windows.
- **Representation:** Each 15-s MIDI window was transformed into a piano-roll matrix of dimension  $800 \times 131$  (128 pitches + 3 pedal channels), with MIDI velocity as a matrix value.
- **Architecture:** The network consists of two residual blocks, each containing two convolution layers. A final projection layer was attached at the end to output the seven-dimension perceptual features.
- **Training:** An Adam optimizer with a learning rate of  $1 \times 10^{-3}$  was used. After 20 epochs, the training converged with a validation loss of 0.038.

## References

1. Widmer, G.; Dixon, S.; Goebel, W.; Pampalk, E.; Tobudic, A. In Search of the Horowitz Factor. *AI Mag.* **2003**, *24*, 111–130.
2. Cancino-Chacón, C.; Peter, S.; Hu, P.; Karystinaios, E.; Henkel, F.; Foscari, F.; Widmer, G. The ACCompanion: Combining Reactivity, Robustness, and Musical Expressivity in an Automatic Piano Accompanist. In Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, Macau, China, 19–25 August 2023. [\[CrossRef\]](#)
3. Morsi, A.; Zhang, H.; Maezawa, A.; Dixon, S.; Serra, X. Simulating and Validating Piano Performance Mistakes for Music Learning Context. In Proceedings of the Sound and Music Computing Conference (SMC), Porto, Portugal, 1–6 July 2024.
4. Aljanaki, A. A data-driven approach to mid-level perceptual musical feature modeling. In Proceedings of the 19th International Society for Music Information Retrieval Conference, (ISMIR), Paris, France, 23–27 September 2018.
5. Chowdhury, S.; Vall, A.; Haunschmid, V.; Widmer, G. Towards explainable music emotion recognition: The route via Mid-level features. In Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, 4–8 November 2019.
6. Maezawa, A.; Yamamoto, K.; Fujishima, T. Rendering music performance with interpretation variations using conditional variational RNN. In Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR), Delft, The Netherlands, 4–8 November 2019.
7. Zhang, H.; Dixon, S. Disentangling the Horowitz Factor: Learning Content and Style From Expressive Piano Performance. In Proceedings of the ICASSP 2023—2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023. [\[CrossRef\]](#)
8. Widmer, G.; Goebel, W. Computational models of expressive music performance: The state of the art. *J. New Music. Res.* **2004**, *33*, 203–216. [\[CrossRef\]](#)
9. Cancino-Chacón, C.E.; Grachten, M.; Goebel, W.; Widmer, G. Computational Models of Expressive Music Performance: A Comprehensive and Critical Review. *Front. Digit. Humanit.* **2018**, *5*, 1–23. [\[CrossRef\]](#)
10. Kirke, A.; Miranda, E. *Guide to Computing for Expressive Music Performance*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013. [\[CrossRef\]](#)
11. Cancino-Chacón, C.E. Computational Modeling of Expressive Music Performance with Linear and Non-Linear Basis Function Models. Ph.D. Thesis, Johannes Kepler University Linz, Linz, Austria, 2018.
12. Jeong, D.; Kwon, T.; Kim, Y.; Lee, K.; Nam, J. VirtuosoNet: A Hierarchical RNN-based System for Modeling Expressive Piano Performance. In Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR), Delft, The Netherlands, 4–8 November 2019.
13. Rhyu, S.; Kim, S.; Lee, K. Sketching the Expression: Flexible Rendering of Expressive Piano Performance with Self-Supervised Learning. In Proceedings of the 23rd International Society on Music Information Retrieval (ISMIR), Bengaluru, India, 4–8 December 2022.
14. Borovik, I.; Viro, V. ScorePerformer: Expressive Piano Performance Rendering with Fine-grained Control. In Proceedings of the 24th International Society on Music Information Retrieval (ISMIR), Milan, Italy, 10–14 November 2023.
15. Peter, S.D.; Cancino-chacón, C.E.; Widmer, G. Sounding Out Reconstruction Error-Based Evaluation of Generative Models of Expressive Performance. In Proceedings of the Digital Libraries for Musicology (DLfM), Milan, Italy, 10 November 2023. [\[CrossRef\]](#)
16. Plasser, M.; Peter, S.; Widmer, G. Discrete Diffusion Probabilistic Models for Symbolic Music Generation. In Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, Macau, China, 19–25 August 2023. [\[CrossRef\]](#)

17. Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; Sutskever, I. Zero-Shot Text-to-Image Generation. In Proceedings of the 38th International Conference on Machine Learning (ICML), Online, 24–26 January 2021.
18. Kong, Z.; Ping, W.; Huang, J.; Zhao, K.; Catanzaro, B. DiffWave: A Versatile Diffusion Model for Audio Synthesis. In Proceedings of the ICLR 2021—9th International Conference on Learning Representations, Vienna, Austria, 3–7 May 2021.
19. Chen, N.; Zhang, Y.; Zen, H.; Weiss, R.J.; Norouzi, M.; Chan, W. WaveGrad: Estimating Gradients for Waveform Generation. In Proceedings of the ICLR 2021—9th International Conference on Learning Representations, Vienna, Austria, 3–7 May 2021.
20. Kim, H.; Kim, S.; Yoon, S. Guided-TTS: A Diffusion Model for Text-to-Speech via Classifier Guidance. In Proceedings of the Machine Learning Research, Baltimore, MD, USA, 17–23 July 2022.
21. Hawthorne, C.; Simon, I.; Roberts, A.; Zeghidour, N.; Gardner, J.; Manilow, E.; Engel, J. Multi-instrument Music Synthesis with Spectrogram Diffusion. In Proceedings of the International Society on Music Information Retrieval (ISMIR), Bengaluru, India, 4–8 December 2022.
22. Mittal, G.; Engel, J.; Hawthorne, C.; Simon, I. Symbolic Music Generation with Diffusion Models. In Proceedings of the 22nd International Society on Music Information Retrieval (ISMIR), Online, 7–12 November 2021.
23. Roberts, A.; Engel, J.; Raffel, C.; Hawthorne, C.; Eck, D. A hierarchical latent vector model for learning long-term structure in music. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 10–15 July 2018.
24. Zhang, H.; Karystinaios, E.; Dixon, S.; Widmer, G.; Cancino-Chacón, C.E. Symbolic Music Representations for Classification Tasks: A Systematic Evaluation. In Proceedings of the 24th International Society on Music Information Retrieval (ISMIR), Milan, Italy, 10–14 November 2023.
25. Cheuk, K.W.; Sawata, R.; Uesaka, T.; Murata, N.; Takahashi, N.; Takahashi, S.; Herremans, D.; Mitsufuji, Y. DiffRoll: Diffusion-Based Generative Music Transcription with Unsupervised Pretraining Capability. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023.
26. Min, L.; Jiang, J.; Xia, G.; Zhao, J. Polyffusion: A Diffusion Model for Polyphonic Score Generation with Internal and External Controls. In Proceedings of the 24th International Society on Music Information Retrieval (ISMIR), Milan, Italy, 5–9 November 2023.
27. Ho, J.; Jain, A.; Abbeel, P. Denoising diffusion probabilistic models. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Online, 6–12 December 2020.
28. Perez, E.; Strub, F.; De Vries, H.; Dumoulin, V.; Courville, A. FiLM: Visual reasoning with a general conditioning layer. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, LA, USA, 2–7 February 2018. [\[CrossRef\]](#)
29. Kim, H.; Serra, X. DiffVel: Note-Level MIDI Velocity Estimation for Piano Performance by A Double Conditioned Diffusion Model. In Proceedings of the 16th International Symposium on Computer Music Multidisciplinary Research (CMMR), Tokyo, Japan, 13–17 November 2023.
30. Ho, J.; Salimans, T. Classifier-Free Diffusion Guidance. In Proceedings of the NeurIPS Workshop on Deep Generative Models and Downstream Applications, Online, 13 December 2021.
31. Gillick, J.; Yang, J.; Cella, C.E.; Bamman, D. Drumroll Please: Modeling Multi-Scale Rhythmic Gestures with Flexible Grids. *Trans. Int. Soc. Music. Inf. Retr.* **2021**, *4*, 156–166. [\[CrossRef\]](#)
32. Cancino-Chacón, C.; Peter, S.D.; Karystinaios, E.; Foscari, F.; Grachten, M.; Widmer, G. Partitura: A Python Package for Symbolic Music Processing. In Proceedings of the Music Encoding Conference (MEC), Halifax, NS, Canada, 19–22 May 2022.
33. Chowdhury, S. Modelling Emotional Expression in Music Using Interpretable and Transferable Perceptual Features. Ph.D. Thesis, Johannes Kepler University Linz, Linz, Austria, 2022.
34. Chowdhury, S.; Widmer, G. On Perceived Emotion in Expressive Piano Performance: Further Experimental Evidence for the Relevance of Mid-Level Perceptual Features. In Proceedings of the Proceeding of the 22nd International Society on Music Information Retrieval (ISMIR), Online, 7–12 November 2021.
35. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. MixUp: Beyond empirical risk minimization. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018—Conference Track Proceedings, Vancouver, NS, Canada, 30 April–3 May 2018.
36. Goebel, W. The Vienna 4x22 Piano Corpus. 1999. Available online: [https://repo.mdw.ac.at/projects/IWK/the\\_vienna\\_4x22\\_piano\\_corpus/index.html](https://repo.mdw.ac.at/projects/IWK/the_vienna_4x22_piano_corpus/index.html) (accessed on 22 July 2024).
37. Peter, S.D.; Cancino-chacón, C.E.; Foscari, F.; Henkel, F.; Widmer, G. Automatic Note-Level Alignments in the ASAP Dataset. *Trans. Int. Soc. Music. Inf. Retr. (TISMIR)* **2023**, *6*, 27–42. [\[CrossRef\]](#)
38. Zhang, H.; Tang, J.; Rafee, S.; Dixon, S.; Fazekas, G. ATEPP: A Dataset of Automatically Transcribed Expressive Piano Performance. In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Bengaluru, India, 4–8 December 2022.
39. Goebel, W. Melody lead in piano performance: Expressive device or artifact? *J. Acoust. Soc. Am.* **2001**, *110*, 641. [\[CrossRef\]](#) [\[PubMed\]](#)
40. Bresin, R.; Umberto Battel, G. Articulation Strategies in Expressive Piano Performance Analysis of Legato, Staccato, and Repeated Notes in Performances of the Andante Movement of Mozart’s Sonata in G Major (K.545). *J. New Music. Res.* **2000**, *29*, 211–224. [\[CrossRef\]](#)
41. Kosta, K.; Bandtlow, O.F.; Chew, E. Dynamics and relativity: Practical implications of dynamic markings in the score. *J. New Music. Res.* **2018**, *47*, 438–461. [\[CrossRef\]](#)

42. Liu, H.; Chen, Z.; Yuan, Y.; Mei, X.; Liu, X.; Mandic, D.; Wang, W.; Plumbley, M.D. AudioLDM: Text-to-Audio Generation with Latent Diffusion Models. In Proceedings of the 40th International Conference on Machine Learning (ICML), Honolulu, HI, USA, 23–29 July 2023.
43. Zhang, Y.; Huang, N.; Tang, F.; Huang, H.; Ma, C.; Dong, W.; Xu, C. Inversion-based Style Transfer with Diffusion Models. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.