



# MuseFlow: music accompaniment generation based on flow

Fanyu Ding<sup>1</sup> · Yidong Cui<sup>1</sup>

Accepted: 23 April 2023 / Published online: 5 July 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

## Abstract

Arranging and orchestration are critical aspects of music composition and production. Traditional accompaniment arranging is time-consuming and requires expertise in music theory. In this work, we utilize a deep learning model, the flow model, to generate music accompaniment including drums, guitar, bass, and strings based on the input piano melody, which can assist musicians in creating popular music. The main contributions of this paper are as follows: 1) We propose a new pianoroll representation that solves the problem of recognizing the onset of a musical note and saves space. 2) We introduce the MuseFlow accompaniment generation model, which can generate multi-track, polyphonic music accompaniment. To the best of our knowledge, MuseFlow is the first flow-based music generation model. 3) We incorporate a sliding window into the model to enable long sequence music generation, breaking the length limitation. Comparisons on datasets such as LDP, FreeMidi, and GPMD verify the effectiveness of the model. MuseFlow produces better results in accompaniment quality and inter-track harmony. Additionally, the note pitch and duration distributions of the generated accompaniment are much closer to the real data.

**Keywords** Accompaniment generation · Flow-based model · New pianoroll · Deep learning

## 1 Introduction

Music composition and production are both professional endeavors. Even an amateur who wishes to write a simple melody should be familiar with music theories such as harmonics, composition, and musicology. The accompaniment arranging (or orchestration) stage in music production transforms music for a single instrument into music that involves a set of performers. With different accompaniment arrangements, a single melody can express entirely different emotions. This stage necessitates a thorough understanding of music theory and, in some cases, the collaboration of multiple professionals, which makes it a mission impossible for an amateur.

In recent years, the idea of creating music by AI (Artificial Intelligence) has attracted extensive attentions. The

related research includes automatic melody generation, automatic orchestration, intelligent lyrics and so on. This research focuses on the generation of accompaniment and tries to make the arranging easier for both professionals and non-professionals. This paper contains a large number of music terms. For the convenience of readers, explanations for most of the terms can be found here [1].

Essentially, without considering the constraints of instrumental diversity in musical orchestration, accompaniment arranging can be deemed as a multi-track generation task. Several deep generation models have been adapted to generate multi-track sequences.

At present, most music accompaniment models, which are mainly based on GAN [2], VAE [3] and Transformer [4] models, have the limitation of long sequence generation problem and the inter-track dependency problem. For more details, please refer to the next section of this paper.

In this paper, we propose a new music representation and music accompaniment generation model to analyze the potential rhythm patterns in melody and learn the correlation between tracks. We generate multi-track accompaniment for the input piano melody, including drums, guitar, bass and strings.

The main contributions of this paper are as follows:

---

✉ Yidong Cui  
cyd@bupt.edu.cn

Fanyu Ding  
nuoyi618@bupt.edu.cn

<sup>1</sup> Beijing University of Posts and Telecommunications, Beijing 100876, China

- We designed a new pianoroll representation, which can solve the problem of recognizing the onset of a musical note in the pianoroll representations as well as saving space.
- We proposed the MuseFlow accompaniment generation model, which can generate multi-track, polyphonic music accompaniment. To the best of our knowledge, MuseFlow is the first Flow-based music generation model.
- We introduced sliding window into the model for long sequence music generation, breaking the length limitation. footnotenote
- Evaluations show that MuseFlow generates higher quality composition than MMM and MuseGAN do.

## 2 Related work

*Generative Adversarial Networks* (GAN) [2] has been widely used since its introduction in 2014. MuseGAN [5] is the first GAN-based model capable of producing multi-track music. MuseGAN provides three generation techniques: interference model, composer model, and mixed model. The team of MuseGAN also proposed another GAN-based model: MidiNET [6], in which the chord information is innovatively added to input data. However, MidiNet employs only 24 notes to represent the melody without any rest.

*Auto-Encoding Variational Bayes* (VAE)[3] is a type of likelihood-based generative model. MusicVAE [7] is a VAE-based model that uses *Recurrent Neural Network* (RNN) structure as “conductor” in the decoder to overcome the problem of long sequence generation. It also uses feature vector operations to control note density and pitch difference.

Microsoft designed a representative multi-track music generation model “XiaoIce Band” [8], which devises a *Chord based Rhythm and Melody Cross Generation model* (CRMCG) and proposes a *Multi-Instrument Co-Arrangement Model* (MICA) using multi-task learning for multi-track music arrangement. Microsoft also designed PopMAG [9] model based on Transformer-XL [10]. However, PopMAG does not avoid the problem of long sequence dependency.

LakhNES[11], a Transformer-XL based model, demonstrates the difficulty in learning the dependence between musical instruments and poor coordinates in the generated results. MMM [12], a GPT2 based model, also has length limitations on bars.

Our proposed model, MuseFlow, is based on the Flow model. Flow model is designed by L.Dinh for NICE [13] in 2015, and was later optimized by Real NVP [14]. In 2018, google created the Glow model [15] to apply the Flow model to face generation. Later NVIDIA applies the Flow model to audio generation, and puts forward the WaveGlow [16]

model to generate speech. Microsoft used the Flow model to detect the video target [17]. MoGlow [18] uses standardized Flow model to realize speech, and then generates human body pose.

The core idea of Flow is to transform the real sample data  $X$  into data  $Y$  which obeys a specified prior distribution, and the transformation function between them is expressed by  $y = f(x)$ . The model is trained by fitting the transformation function  $x = f^{-1}(y)$  with neural networks, and its solution is derived from the variable replacement Equation (1) of two PDFs(Probability Density Functions), where  $\frac{\partial f(x)}{\partial x}$  is the Jacobian matrix of the function  $f$  at  $x$ , and  $det$  represents the determinant.

$$p_X(x) = q_\theta(f(x)) \left| det \frac{\partial f(x)}{\partial x} \right| \quad (1)$$

The generative procedure is an estimation of the probability distribution with parameters  $\theta$ . In general, the maximum likelihood method is used to compute  $\theta$ , so the goal of the Flow algorithm optimization is to maximize Equation (2).

$$\log(p_X(x)) = \log(q_\theta(f(x))) + \log(|det \frac{\partial f(x)}{\partial x}|) \quad (2)$$

The difficulty in solving the equation lies in the need for a known inverse function to calculate the Jacobian determinant. This requirement mandates that the transfer function  $f$  must be a continuous, differentiable, and reversible nonlinear function. However, the Flow model’s clever design solves these issues. For more details, please refer to [13].

The Flow model’s key characteristic is its conversion reversibility. It enables transformation from  $X$  distribution to  $Y$  distribution and from  $Y$  distribution to  $X$  distribution simultaneously. In other words, the Flow model solves the two-way channel of two distributions, implying that the dimensions of  $X$  and  $Y$  must be the same.

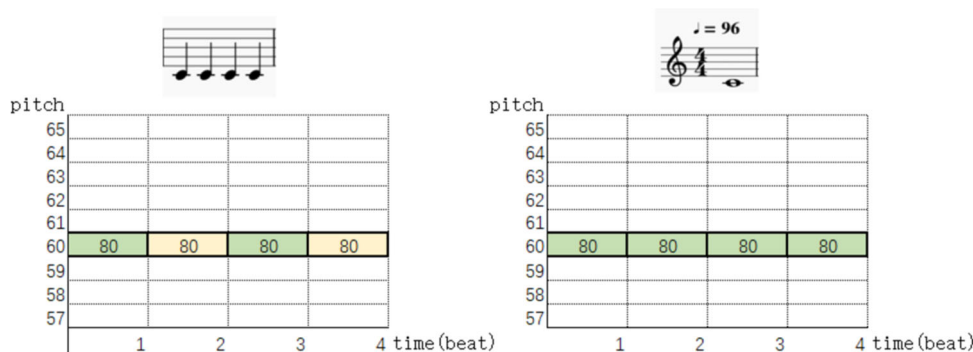
## 3 Proposed model

Choosing an appropriate music data representation is crucial for the model’s performance. In this section, we first introduce the new pianoroll representation and then discuss the MuseFlow model structure and optimization scheme.

### 3.1 Data representation

The pianoroll is a music representation that uses a two-dimensional matrix to store information, with time and pitch represented on the horizontal and vertical axes, respectively. The corresponding values in the matrix indicate the velocity of each music note. This allows for the representation of multi-track accompaniment in a single matrix. However, the

**Fig. 1** Example of pianoroll representation



pianoroll representation encounters difficulties when there are consecutive identical music notes in the score. Figure 1(a) shows an example where four adjacent quarter notes of different colors cannot be distinguished from one whole note when converted into the pianoroll representation. To overcome this challenge, we propose a new representation.

The proposed new representation, called the new pianoroll representation (NPR), stores each music note using an 8-bit binary byte (uint8), while maintaining the same semantics for the horizontal and vertical coordinates. The vertical axis represents pitch, the horizontal axis represents time, and the numerical values indicate velocity. In addition, we designed three states of velocity information to distinguish onset notes.

1. No sound or zero velocity: 0000 0000 = 0
2. Delay (i.e. the sound after pressing the key): 0000 0001 0111 1111 = [1,127]
3. Start (the first position of the sound): 1000 0001 1111 1111 = [129,255]

The velocity range of the NPR is from 0 to 127, which uses only half of the uint8 range from 0 to 255. A velocity value of 0 indicates no sound. The values of state 1 and state 2 represent the actual velocity values. The value of state 3 is obtained by inverting the original velocity value, i.e., [-127,-1], which is then represented using uint8 as [129,255], and corresponds to onset notes. Figure 1(b) provides an example of the NPR representation.

NPR marks the onset of a note and saves space, especially when training the model using GPU, as it reduces memory usage. To determine if the data is an onset note, we only need to perform a logical AND operation with 1000000. To obtain the original velocity value, we only need to perform a logical AND operation with 01111111.

NPR may improve the quality of the generated music segments, as shown in Table 2 and Table 4.

### 3.2 MuseFlow

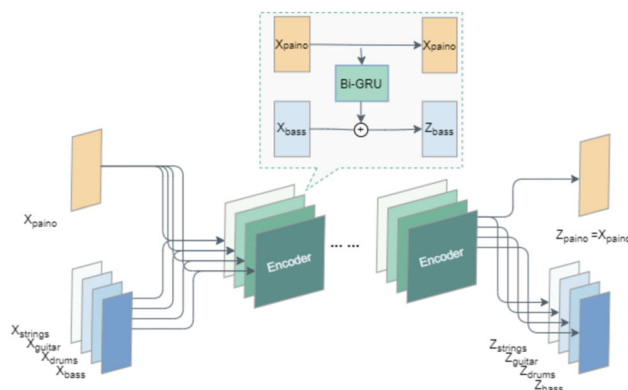
The MuseFlow is a deep learning model based on Flow, used for generating music accompaniment. It possesses a nice fea-

ture: forward training is the encoding process, and music materials are mapped to multi-dimensional Gaussian space by multiple encoders; the reverse operation is the generation, i.e., decoding process, where the trained encoders are generators. This section describes the encoder and generator structures of MuseFlow, as well as further optimization of the model structure in experiments.

#### 3.2.1 Encoders

To capture the diverse characteristics of different musical tracks, MuseFlow utilizes multiple encoders with distinct parameters but the same internal structure. For training, MuseFlow used five-track pianorolls (shown in Fig. 2), which included piano, drums, guitar, bass, and strings. The piano track serves as the guiding melody for the other tracks, while the accompaniment tracks are transformed by multiple encoders, with each dimension mapped to a Gaussian distribution. The encoder output, represented as the hidden variable Z, is obtained for all five tracks, including the piano melody and the four accompanying tracks. This Z vector is then used to generate a new set of pianorolls for the accompanying tracks.

Compared with Flow, the structure of MuseFlow incorporates two main improvements.



**Fig. 2** MuseFlow training process

1. MuseFlow simplifies the structure by eliminating the need to exchange constant parts between encoders. Additionally, the main melody track remains unchanged and is used as lead data.
2. MuseFlow enables parallel mapping of multiple channels by processing all four accompaniment tracks simultaneously.

The calculation process for each encoder is shown in Equation (7), where  $x_{piano}$  represents the input piano melody and  $x_{other}$  represents the input accompaniment tracks (drums, guitar, bass, strings). The output feature vector for the  $i$ -th encoder is represented by  $z_{i,other}$ , with  $z_{0,other} = x_{other}$ . The transformation function  $An$  is implemented using a neural network, with MuseFlow using a Bi-directional Gated Recurrent Unit (Bi-GRU) [19]. At each time step, the pitch at the current time point is used as input to the Bi-GRU.

$$\begin{cases} x_{piano} = x_{piano} \\ z_{i, other} = g_n^{-1}(z_{i-1,other}, An(x_{piano})) \end{cases} \quad (3)$$

$$z_{n, other} = g_n^{-1} \circ \dots \circ g_1^{-1}(x_{other}) \quad (4)$$

The  $x_{piano}$  has two paths in the encoder: one path is output directly as part of the next encoder input; the other path,  $x_{piano}$  enters the neural network  $An$  to obtain  $An(x_{piano})$ , which has the same dimension as  $x_{other}$ . The vectors  $An(x_{piano})$  and  $z_{i-1,other}$  are input to the linear function  $g_n^{-1}$  to obtain  $z_{i,other}$ . After merging  $x_{piano}$  and  $z_{i,other}$ , the resulting vector is then input to the next encoder, and so on, through  $n$  encoders. As shown in Equation (8),  $z_{n,other}$  represents the final feature vector of the accompaniment track output by the model after  $n$  encoders.

### 3.2.2 Generators

The generation process of MuseFlow is illustrated in Fig. 3. The generator takes as input the piano track  $x_{piano}$  and a random vector  $z_{n,other}$  sampled from a Gaussian distribution, which has the same tensor size as the tracks to be generated. As in the training phase, the piano track  $x_{piano}$  is first converted into a feature vector  $An(x_{piano})$  by the Bi-GRU

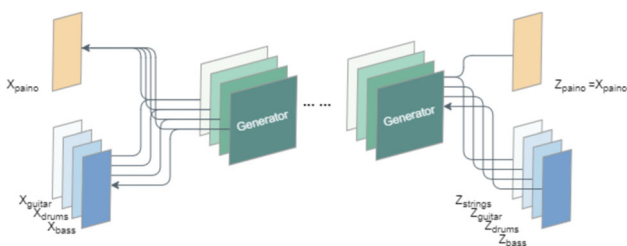


Fig. 3 MuseFlow generation process

network.  $An(x_{piano})$  and the random vector  $z_{n,other}$  are then fed into the function  $g_n$  to obtain  $z_{i-1,other}$ , where  $g_n$  and  $g_n^{-1}$  are inverse functions. The resulting vector  $z_{i-1,other}$  is combined with  $x_{piano}$  and input to the next generator, and the process is repeated for  $n$  generators. Finally, the output is the pianoroll with five tracks, where  $x_{other}$  represents the generated accompaniment tracks.

$$\begin{cases} x_{piano} = x_{piano} \\ z_{i-1, other} = g_n(z_{i,other}, An(x_{piano})) \end{cases} \quad (5)$$

$$x_{other} = g_1(g_2(\dots g_n(z_{n,other})) \quad (6)$$

### 3.2.3 Sliding generation

The Flow-based model requires input and output dimensions to be equal, which can consume a large amount of GPU memory during training and limit the length of generated music segments. To address this issue, we propose a sliding window scheme to aid in training and generation. Specifically, we set the size of the sliding window to be the length of the music segment (eight bars in our work). The encoder employs a bidirectional gated recurrent unit (Bi-GRU) to reduce discontinuity in the melody caused by the segmentation. When dividing the music data into segments, we move forward by only four bars each time. Therefore, the generated result is influenced by the adjacent music segment in the input data. The sliding generation process is illustrated in Fig. 4, where  $Z_i$  represents the random vectors of four bars and  $X_i$  represents the generated result of four bars.  $G$  represents the MuseFlow generator. Except for segments  $X_1$  and  $X_2$ , the subsequent generation process is as follows: (1) The  $G$  generator converts the two random variables  $[Z_i, Z_{i+1}]$  into  $[X_i, X_{i+1}]$ ; (2) The generated sequence  $X_{i+1}$  is spliced with the previous sequence  $[X_1, X_2, \dots, X_{i-1}]$ . This design

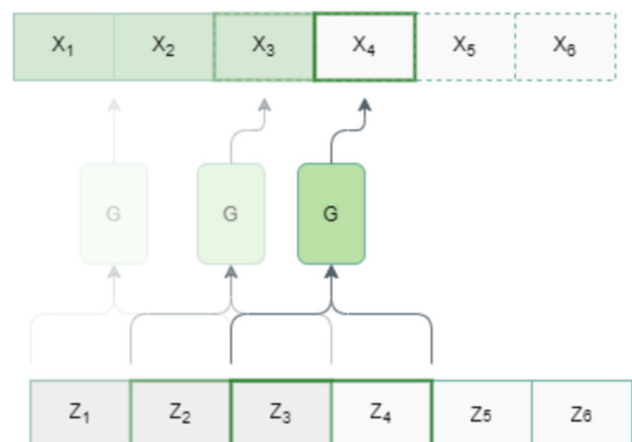


Fig. 4 Sliding generation

**Table 1** Statistics for the datasets

Dataset	MusicalPieces
LPD	35396
Freemidi	19271
GPMD	22130

helps to generate longer music sequences under limited GPU memory and computational power. However, we must point out that a potential issue is that the randomness of vector  $Z_i$  may lead to an unsmooth connection between two adjacent music segments.

## 4 Experimental Setup

In this paper, we employ several datasets to demonstrate the validity of our model. In this section, we first describe the datasets and data processing procedures. Then, we discuss the comparison with other models.

### 4.1 Data Preprocessing

The experiment utilized three datasets: the Lakh Pianoroll Dataset (LPD)[20], Freemidi<sup>1</sup>, and our collection of global popular music datasets (GPMD). The original MIDI files in these datasets contained significant noise and required extensive data processing. After data cleaning and filtering, we obtained the processed data shown in Table 1. For testing, 1000 pieces were randomly extracted from these datasets, while the remaining data were used for training.

The data processing procedures are as follows:

1. Data Cleaning: We removed tracks with few notes or empty tracks. Specifically, we discarded tracks whose sum of notes was less than 10
2. Melody Identification: If the MIDI file had a marked melody, we used the marked track directly. If there was no mark, we used the MIDI Miner[21] random forest model to identify the main melody. Any unrecognized MIDI file was discarded.
3. Data Filtering: We filtered the MIDI files using the following conditions:
  - (a) 4/4 beat (or tempo).
  - (b) There are five tracks including drums, guitar, bass, strings and piano, with at least three tracks present.
  - (c) Piano as the main melody.
  - (d) If the same instrument had more than one track, we sorted them based on their duration and kept the longest track.

<sup>1</sup> <https://freemidi.org/genre-pop>

4. Music Segmentation: We converted the music sequences of different lengths into pianorolls and sliced them with uniform length. In the experiment, we used 8 bars as one segment, with 4 beats in each bar. Furthermore, we divided each beat into 24 time slots, resulting in a time dimension of  $24 * 4 * 8 = 768$  for each music segment. We observed that the common pitch range was 22-88 (i.e., 60 pitches in total). Therefore, the total data dimension of a music segment was  $768 * 60 * 5$ .

### 4.2 Model training

The MuseFlow model had four encoders, and the additive coupling layer was used inside. The mapping process of a function was as follows: the input  $x_{piano}$  first passed through the Normalization Layer, to prevent over-fitting, then entered the Bi-GRU network with 128 neural units. It outputted the hidden state of each time step and passed through the fully connected neural network with dimension 60, to ensure that the tensor dimensions of  $An(x_{piano})$  and  $x_{other}$  were the same in the coupling operation. We trained the model on an NVIDIA 3090 GPU, using the Adam optimizer, log-likelihood loss function, and batch size of 128.

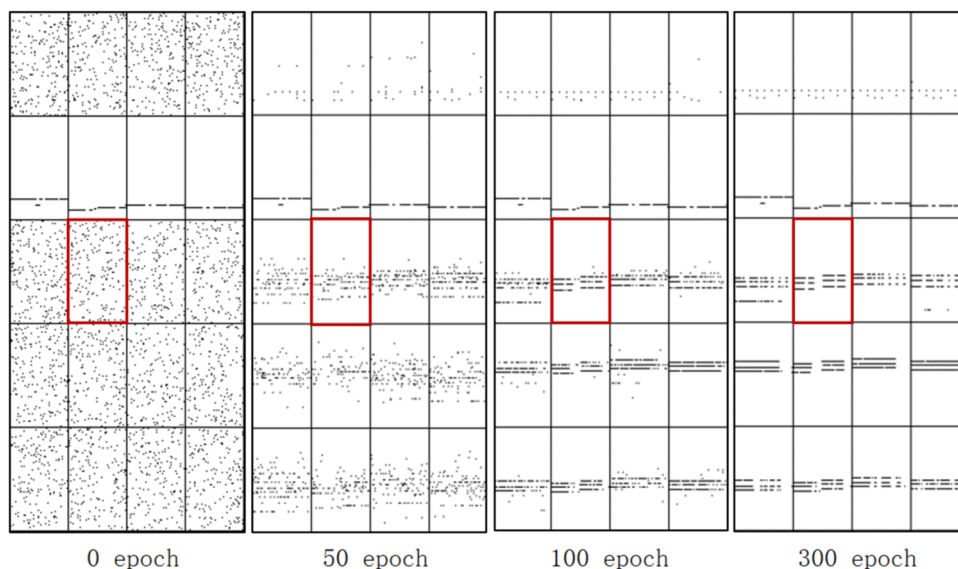
In Fig. 5, we show the results of the MuseFlow model at different stages of training, with tracks for drums, piano, guitar, bass, and strings. The model generates music segments of 8 bars, with 4 beats in each bar and each beat divided into 24 time slots. The x-axis represents time, and the y-axis represents pitch. As the training epoch increases, the generated music becomes clearer and more coherent. Figure 5(b) shows a zoomed-in detail of the guitar track in (a), highlighted by a red border.

### 4.3 Evaluation Metrics

The quality of music is inherently subjective, but we can use certain measurements to evaluate it. Inspired by Yang and Lerch [22], Harte et al. [23], Dong et al. [5], we have selected the following metrics to assess the quality of our music accompaniment generated using the models:

- Pitch Classes (PC): This measures the number of used pitch classes per bar, on a scale of 0 to 12.
- Pitch Shift (PS): This measures the average interval between two consecutive pitches, in semitones.
- Inter-onset-interval (IOI): This measures the time between two consecutive notes.
- Polyphony Rate (PR): This measures the ratio of time steps where multiple pitches are played simultaneously. We ignore the drum track in this calculation.

**Fig. 5** MuseFlow training results



- **Tonal Distance (TD):** This measures the harmonicity between a pair of tracks. Smaller TD values indicate greater harmonicity.

In addition, we also evaluated the distribution of pitch and note duration. To do so, we used the following metrics:

- **Note Duration Histogram (NDH):** This calculates the distribution of note duration across 12 categories based on the beat, including full, half, quarter, 8th, 16th, dot half, dot quarter, dot 8th, dot 16th, half note triplet, quarter note triplet, and 8th note triplet.
- **Pitch Class Histogram (PCH):** This is an octave-independent representation of the pitch content, with a dimensionality of 12 for a chromatic scale.

Our evaluation process involves three steps. First, we calculate the pitch histogram and note duration histogram metrics for all generated results. Second, we use kernel density estimation to convert each histogram into a probability distribution function. Finally, we calculate the coverage area between the resulting distribution and the ground-truth distribution using Equation (11).

$$D_x = \frac{1}{N_{tracks} * N_{bars}} \sum_{i=1}^{N_{tracks}} \sum_{j=1}^{N_{bars}} OA(P_{i,j}^x, \hat{P}_{i,j}^x) \quad (7)$$

Where,  $x$  represents the pitch histogram or the note duration histogram;  $P_{i,j}^x$  represents the  $x$  metric in the ground-truth set;  $\hat{P}_{i,j}^x$  represents the generated data;  $OA$  represents the Overlapping Area of two PDFs (Probability Distribution Function).

## 4.4 Results and analysis

To evaluate these models, we randomly selected 1,000 music pieces from the LPD, FreeMidi, and GPMD datasets as test cases. We used the MuseGAN, MMM, and MuseFlow models to generate accompaniments for drums, guitar, bass, and strings based on the main piano melody. To verify the usefulness of NPR, we used MuseFlow to represent the effect of the model without NPR, and MuseFlow+NPR to represent the effect of using NPR. Ground-Truth represents the evaluation results of the true music data used for reference.

### 4.4.1 Intra-track quality

Table 2 shows the distances between the generated results and the Ground-Truth, with Ground-Truth serving as the baseline. In terms of intra-track metrics, MuseFlow is closer to the real accompaniment situation compared to the MuseGAN and MMM models. Moreover, MuseFlow+NPR, which marks the onset point of notes to retain the original information of the music and avoid confusion between continuous notes and long notes, further improves the generation quality compared to MuseFlow itself.

### 4.4.2 Inter-track quality

The results of the inter-track metrics are presented in Table 3, which compares the relationship between the piano and any of the other four tracks (P: piano, D: drums, G: guitar, B: bass, S: string) by calculating the TD distance. Among the three models, MuseFlow shows relatively stronger harmonization. It was observed that the new pianoroll representation (NPR) has almost no effect on the inter-track distance in the experiments, so we did not include its own comparison results.

**Table 2** Intra-track evaluation (the best indicators are marked in bold)

	DataSet	PC	PS	IOI	PR
LDP	Ground-Truth	3.546	3.969	2.329	0.434
	MuseGan	+4.750	+12.148	-0.848	+0.557
	MMM	+0.758	+1.299	+0.278	-0.180
	MuseFlow	-0.479	-1.092	+0.201	-0.167
	<b>MuseFlow+NPR</b>	<b>-0.472</b>	<b>-1.009</b>	<b>+0.109</b>	<b>-0.167</b>
FreeMidi	Ground-Truth	3.624	4.873	2.471	0.454
	MuseGan	+4.524	+7.166	-1.019	+0.536
	MMM	+0.993	+1.181	-0.389	+0.196
	MuseFlow	-0.423	-1.160	-0.171	-0.155
	<b>MuseFlow+NPR</b>	<b>-0.416</b>	<b>-1.133</b>	<b>-0.117</b>	<b>-0.153</b>
GPMD	Ground-Truth	3.624	4.873	2.471	0.454
	MuseGan	+4.477	+6.711	-0.942	+0.600
	MMM	+0.729	+3.525	-0.163	-0.192
	MuseFlow	-0.665	-1.798	-0.046	-0.155
	<b>MuseFlow+NPR</b>	<b>-0.454</b>	<b>-1.609</b>	<b>-0.041</b>	<b>-0.088</b>

This is because while the NPR representation improves the intra-track storage quality, it does not affect the inter-track relationship. Generally, the results generated by MuseGAN are quite different from the real dataset, especially in terms of pitch shift. On the other hand, the results generated by MMM exhibit higher intra-track quality but lack good harmony among the tracks. In comparison, MuseFlow performs better in both intra-track and inter-track metrics.

#### 4.4.3 The distribution of note pitch and note duration

The distribution of pitch and note duration can be crucial features of a melody. In our evaluation, we calculated the pitch histogram and note duration histogram metrics for all generated results. We then applied kernel density estimation to

convert the histograms into probability distribution functions and calculated the overlapping area with the ground-truth distribution using Equation (11).

Table 4 shows the results of the overlapping area (OA) calculations, where MuseFlow achieved the highest OA values, indicating that it is closer to the real data compared to the other models. Furthermore, the new pianoroll representation (NPR) method improves the similarity of the generated data to the real data to some extent.

Figure 6 displays the probability distribution of note duration (a) and pitch (b) for each model. The accompaniments generated by the MMM model exhibit large variances in pitch and duration, with the mean values deviating slightly from the real data. Conversely, the mean values of the MuseGAN model are close to those of the real dataset, but the variances are too small. The use of the NPR representation has

**Table 3** Inter-track evaluation (smaller values are better)

	DataSet	P-D	P-G	P-B	P-S	B-G	B-S
LDP	Ground-Truth	1.568	0.957	1.612	0.750	1.152	1.003
	MuseGan	0.937	1.188	1.254	1.180	0.835	1.035
	MMM	1.514	1.409	1.596	1.172	1.740	1.718
	<b>MuseFlow</b>	<b>0.936</b>	<b>0.644</b>	<b>1.063</b>	<b>0.793</b>	<b>0.607</b>	<b>0.688</b>
FreeMidi	Ground-Truth	1.645	1.045	1.524	0.423	1.211	0.507
	MuseGan	0.738	1.194	1.318	1.285	0.843	1.048
	MMM	1.588	1.301	1.778	1.495	1.571	1.853
	<b>MuseFlow</b>	<b>0.646</b>	<b>0.732</b>	<b>0.974</b>	<b>0.999</b>	<b>0.785</b>	<b>0.774</b>
GPMD	Ground-Truth	1.538	0.777	1.688	0.772	1.017	0.988
	MuseGan	0.779	1.194	1.318	1.285	0.843	1.048
	MMM	1.473	1.274	1.385	1.126	1.460	1.566
	<b>MuseFlow</b>	<b>0.749</b>	<b>0.731</b>	<b>0.898</b>	<b>0.817</b>	<b>0.614</b>	<b>0.683</b>

**Table 4** OA distance on NDH and PCH (larger values are better)

	MuseGan	MMM	MuseFlow	MuseFlow+NPR
NDH	0.751	0.717	0.889	<b>0.902</b>
PCH	0.768	0.405	0.932	<b>0.936</b>

a positive effect on the model generation, and the difference between the two data distributions is not significant.

In summary, the pitch and note duration distributions are important features in evaluating the quality of generated music. The results show that MuseFlow outperforms the other models in terms of pitch and duration distributions, and the NPR representation further improves the generation quality.

#### 4.4.4 The sliding window

The experiment conducted aimed to demonstrate the effectiveness of using sliding windows in long sequence music generation. Three models were trained with different training data segmentation: (a) 2 bars without sliding window, (b) 4 bars without sliding window, and (c) 4 bars with sliding window. The models were evaluated by generating accompaniment for Beethoven's "Ode to Joy" piano melody.

The results are shown in Fig. 7, where the generated tracks are represented by small rectangles, and the second track (from top to bottom) represents the input piano melody. Subfigure (a) shows the generated results without sliding window, where the drum and guitar tracks break obviously between the two segments (bar[1-2] and bar[3-4]). In subfigure (b), where the models were trained with 4 bars without sliding window, there is a better continuity within the internal bars of bar[1-4] and bar[5-8], but the continuity between adjacent bar segments is poor, especially in the bass and

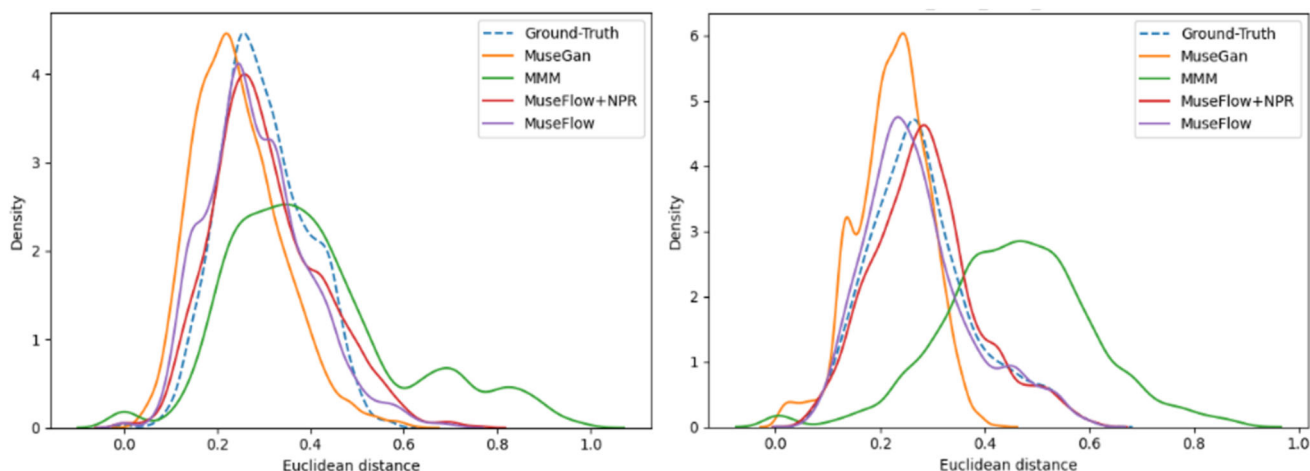
string tracks. This suggests that the model may learn the melody patterns and generate accompaniment within one whole segment, which also means the longer the segment, the greater the consistency of the accompaniment. In subfigure (c), with the use of sliding windows, the adjacent segments show apparent continuity, denoted by the red box. This suggests that sliding windows can help the model generate longer accompaniment by building connections between two segments.

Overall, the results suggest that the use of sliding windows can be an effective technique for improving the continuity and consistency of long sequence music generation.

#### 4.4.5 The robustness of the model

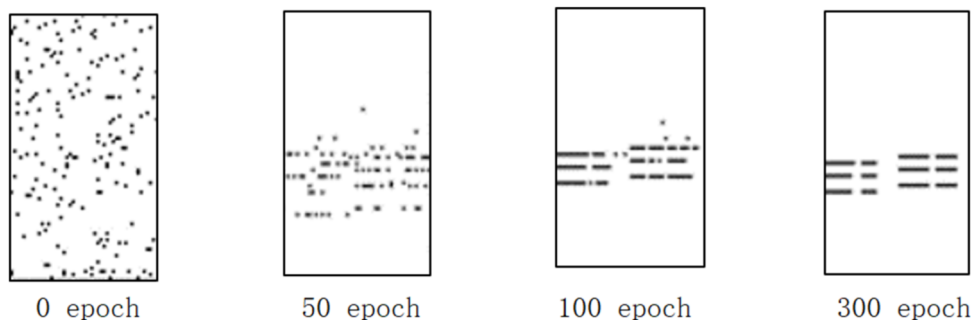
The robustness of the model can be evaluated in various ways. One approach is to train the model on one dataset and test it on another to assess its robustness. We trained MuseFlow, MMM, and MuseGAN models on three datasets, resulting in nine trained models. We randomly selected 1000 samples from each dataset and evaluated the trained models' performance. Without loss of generality, approximately 2/3 of the samples in the test set are not included in the training set and can be regarded as reflecting the model's robustness to a certain extent. We roughly used the deviation of evaluation metrics on three models trained on three datasets (e.g., training MuseFlow on three datasets resulted in MuseFlow1, MuseFlow2, and MuseFlow3 models). Using the same test set to generate outputs on these three models and calculating the standard deviation of the evaluation results represents the deviation to measure the model's robustness.

As shown in Table 5, in the comparison of Intra-track metrics, except for IOI, the MuseFlow model has the smallest variation in evaluation metrics across the three datasets,

**Fig. 6** Probability distribution of the note duration and pitch



**Fig. 7** Track continuity with or without sliding window



**Table 5** Intra-track variation

	PC	PS	IOI	PR
MuseGan	0.146	3.016	<b>0.086</b>	0.033
MMM	0.145	1.321	0.339	0.221
MuseFlow	0.127	0.389	0.189	<b>0.007</b>
MuseFlow+NPR	<b>0.029</b>	<b>0.317</b>	0.115	0.042

**Table 6** Inter-track variation

	P-D	P-G	P-B	P-S	B-G	B-S
MuseGan	0.105	<b>0.003</b>	<b>0.037</b>	<b>0.061</b>	<b>0.005</b>	<b>0.008</b>
MMM	<b>0.058</b>	0.071	0.197	0.201	0.141	0.144
MuseFlow	0.147	0.051	0.083	0.113	0.101	0.051

indicating that its robustness is better than MuseGAN and MMM.

As shown in Table 6, in the comparison of Inter-track metrics, except for P-D, the variation in metrics of the MuseFlow model is moderate, indicating that its robustness is worse than MuseGAN but significantly better than MMM.

## 5 Conclusion

In this paper, we propose a new pianoroll representation that can mark the onset point of notes. Our experiments demonstrate that this representation also improves the realism of generated accompaniment. Additionally, we propose the MuseFlow model for accompaniment generation, which enhances the quality and track harmony of multi-track accompaniment generation. We introduce a sliding window into the model to provide a feasible solution for long-time sequence generation.

To validate our model, we collect and process three music datasets: LPD, FreeMidi, and GPMD. We extract test cases to evaluate MuseFlow, MuseGAN, and MMM models using the same quantifiable evaluation indexes. The results indicate that MuseFlow outperforms MuseGAN and MMM in terms

of pitch and duration distribution, intra-track continuity, and inter-track harmony.

The limitations of this study are the inherent dimensionality waste of the Flow model, which requires extensive computational space during training. On an Nvidia 3090 graphics card, the maximum segment length for training data is 8 bars, making it challenging for the model to learn music patterns longer than 8 bars. However, most popular 4/4 tempo music exceeds 40 bars in length, which demands a high-performance training environment.

There are two future research directions:

1. Explore new music encoding methods that can accommodate the expression of multi-track music, reduce the space occupied by the encoding, and address the problem of dimensionality waste.
2. Consider adopting the channel swapping technique used in the Glow model to enhance the coordination between accompaniment tracks and improve the overall harmony of multi-track accompaniments.

**Data Availability** The source codes of the model introduced in this paper are publicly available at <https://github.com/nuoyi-618/MuseFlow>.

## References

1. Adams, R.: MusicTheory.net. Accessed: 2023-01-10 (2002–2022). <https://www.musictheory.net>
2. Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., Bharath, A.A.: Generative adversarial networks: An overview. *IEEE Signal Processing Magazine* **35** (2018)
3. Kingma DP, Welling M (2014) Auto-encoding variational bayes. Banff, AB, Canada
4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need **30** (2017)
5. Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., Yang, Y.-H.: Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI Press, New Orleans, Louisiana, USA (2018)

6. Yang L-C, Chou S-Y, Yang Y-H (2017) Midinet: A convolutional generative adversarial network for symbolic-domain music generation. Suzhou, China, pp 324–331
7. Roberts, A., Engel, J., Raffel, C., Hawthorne, C., Eck, D.: A hierarchical latent vector model for learning long-term structure in music **80** (2018)
8. Zhu H, Liu Q, Yuan NJ, Qin C, Li J, Zhang K, Zhou G, Wei F, Xu Y, Chen E (2018) Xiaoice band: A melody and arrangement generation framework for pop music. London, United kingdom, pp 2837–2846
9. Ren, Y., He, J., Tan, X., Qin, T., Zhao, Z., Liu, T.-Y.: Popmag: Pop music accompaniment generation. In: Proceedings of the 28th ACM International Conference on Multimedia, pp. 1198–1206. Association for Computing Machinery, New York, NY, USA (2020)
10. Dai, Z., Yang, Z., Yang, Y., Carbonell, J.G., Le, Q., Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 2978–2988 (2019)
11. Donahue C, Mao HH, Li YE, Cottrell GW, McAuley J (2019) Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training. Delft, Netherlands, pp 685–692
12. Ens, J., Pasquier, P.: MMM: Exploring conditional multi-track music generation with the transformer (2020)
13. Dinh L, Krueger D, Bengio Y (2015) Nice: Non-linear independent components estimation. San Diego, CA, United states
14. Dinh L, Sohl-Dickstein J, Bengio S (2017) Density estimation using real nvp. Toulon, France
15. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS 2018), vol. 31 (2018)
16. Prenger, R., Valle, R., Catanzaro, B.: Waveglow: A flow-based generative network for speech synthesis, vol. 2019-May. Brighton, United kingdom, pp. 3617–3621 (2019)
17. Zhu, X., Xiong, Y., Dai, J., Yuan, L., Wei, Y.: Deep feature flow for video recognition, vol. 2017-January. Honolulu, HI, United states, pp. 4141–4150 (2017)
18. Henter, G.E., Alexanderson, S., Beskow, J.: Moglow: Probabilistic and controllable motion synthesis using normalising flows. *ACM Trans. Graph.* **39** (2020)
19. Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., Xu, B.: Attention-based bidirectional long short-term memory networks for relation classification. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* **2** (2016)
20. Raffel, C.: Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching (10134335), 222 (2016)
21. Guo, R., Herremans, D., Magnusson, T.: Midi miner - a python library for tonal tension and track classification (2019)
22. Yang L-C, Lerch A (2020) On the evaluation of generative models in music. *Neural Computing and Applications* 32(9):4773–4784
23. Ramoneda, P., Bernardes, G.: Revisiting harmonic change detection. In: Proceedings of the 149th Audio Engineering Society Convention 2020, Virtual, Online (2020)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.