Cluster Expressive Timing in Performed Music with VQ-VAE

Zihan Chai[1] and Shengchen Li[1][0000-0002-2488-298X]

Xi'an Jiaotong-Liverpool University, No. 111, Ren'ai Road, Suzhou, China
{zihan.chai23@student.xjtlu.edu.cn, shengchen.li@xjtlu.edu.cn}

Abstract. There are many attempts on clustering expressive timing in performed classical piano music which suffer from variable lengths of phrases. This work uses VQ-VAE, a deep learning based method, to cluster expressive timing. The proposed method uses a codebook with a codec structure, where each code vector corresponds to a cluster. The code vectors that is very similar could be further merged, which gives a more flexible way to determine the number of clusters for expressive timing. To evaluate the proposed method, a model selection test with Gaussian Mixture Model (GMM) for expressive timing is repeated to compare the optimal number of clusters in expressive timing. The JS divergence between clusters resulted by both VQ-VAE and GMM is also tested to show the difference of cluster distribution. The result shows that the result of clustering by VQ-VAE is comparable with the results by GMMs.

Keywords: Expressive Timing · Performed Piano Music · Phrasing · VQ-VAE · Clustering

1 Introduction

In computational musicology, clustering expressive timing is one of the common ways to analyse expressiveness as demonstrated by Demos et al. [1]. Though the clustering of intra-phrase expressive timing proposed by Li et al. [2] enables the clustering over variable lengths of phrases. Many empirical parameters such music structure, the selection of music unit for analysis and the number of clusters precent automatic clustering of expressive timing. This paper aims to produce a deep learning based solution that has the potential for automatic expressive timing clustering that is capable of variable phrase lengths and flexible cluster number with the potential of automatic phrasing.

Considering the expressive timing information is usually simple but with a very complex distribution and the computational musicology requires interpretability, VQ-VAE [3] is chosen amongst other deep learning-based clustering

Proc. of the 17th Int. Symposium on Computer Music Multidisciplinary Research, London, United Kingdom, 2025

method due to better interpretability, stability and simplicity. The VQ-VAE has a codec structure that enables proper structures to capture temporal dependencies in expressive timing within a phrase. Unlike traditional VAE [4], the latent space in VQ-VAE is quantised and is represented by a codebook. The expressive timing is firstly encoded as a codebook and then mapped to a code in the codebook for clustering. The codes in codebook may be further merged with a given threshold of distance, which provides flexibility of cluster numbers.

To validate the proposed model, a dataset combing the public shared Mazurka dataset and a private Islamey dataset are used for the proposed experiments. With the dataset, we firstly test the proposed model with different configurations of codebook in the latent space. The experiment reveals how the lengths of code and the size of codebook impact the number of clusters resulted.

To further validate the resulted clusters of expressive timing, the distribution of the resulted cluster is compared. Given the cluster distribution resulted by GMM models and the proposed VQ-VAE models, the difference between resulting distributions is measured by Jensen-Shannon Divergence (JS Divergence) [5].

2 Background

2.1 The Evolution of Expressive Timing Analysis

Expressive timing analysis aims to understand how expressiveness formed by continuous variations in the rhythm of a musical event [6]. Though there are a few attempts that synthesis expressiveness in the domain of waveform directly [7,8], the knowledge of expressiveness in performance is still able to provide expressiveness control that the end-to-end system [9].

The field subsequently shifted towards data-driven methods [10], where clustering is commonly used as a way to analyse expressive timing [1]. Legend analysis methods such Principle Component Analysis [11], Gaussian Mixture Model (GMM) [2] and Self-Organising Map (SOM) [12] have been used for expressive timing analysis.

Existing clustering methods for expressive timing suffer from a few more issues: the number of clusters, the variable lengths of phrases and the unit of analysis.

First, the number clusters is a empirical decision in most studies. Though the model selection test [13] may give objective answer to the optimal number of clusters, the model selection test requires a massive training process, which could be time-consuming and computational expensive. As a result, the proposed system for clustering expressive timing should produce a flexible number of clusters for different datasets.

Second, given the phrase lengths is changed throughout a piece of music, the proposed system should be capable of processing phrase with different lengths. Though Li et al. [2] has provided a solution that regressing intra-phrase expressive timing with polynomial functions, the optimised order of polynomial functions is phrase-dependent. As a result, the proposed system should be able to adapt the inconsistent phrase lengths throughout the pieces of music.

Finally, as expressive timing is associated with music structure [14], it is hard to determine the most proper unit for analysis. As a result, the music structure is considered as an empirical decision for analysis in expressive timing. A possible solution for the problem is automatic phrasing which may be achieved by multi-task learning. As a result, the proposed system should be capable of multi-task learning in the future, which requires a deep learning-based system.

2.2 Deep Generative Models for Unsupervised Clustering

Deep learning techniques has been used for clustering sequential data [15]. Given the case of clustering expressive timing, which is represented as a vector of tempi, the proposed clustering algorithm needs to address the issue of low data availability, low dimensionality and complex distribution. As a result, a generative deep learning model needs to be selected.

Among generative models, the generative adversarial network based solutions such as ClusterGAN [16] may not work well with low data availability. Traditional deep embedding for clustering [17] has low interpretability even with the sequence-to-sequence framework [18] used. As a result, this paper selects Variational Auto-Encoder (VAE) [4] as the framework.

A standard VAE, which uses a simple unit Gaussian prior over the latent space, is not explicitly optimised for discovering cluster structure. Vector-Quantised VAE (VQ-VAE) [3] quantise the feature vector in the latent spaces as a codebook. The quantised feature vectors,can be further merged if necessary.

VQ-VAE has demonstrated its effectiveness in capturing complex musical features. For example, the Jukebox model by Dhariwal et al. employed [19] VQ-VAE to generate high-fidelity music in various genres, showcasing its capability to handle raw audio and symbolic music data. Moreover, VQ-VAE also demonstrates the ability to model discrete latent spaces for sequential data by time-series generation [20].

As a result, this paper proposes VQ-VAE for the study.

3 Methods

3.1 Data Preparation

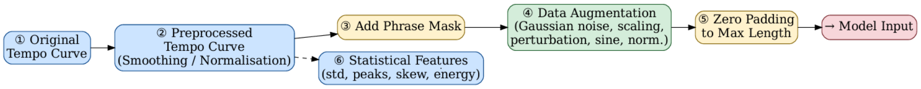


Fig. 1: Data preprocessing that converts original tempo curves to the input of the system.

The expressive timing is represented in the form of tempo curves as the previous work [2]. However, to make the proposed model being capable of potential developments for automatic phrasing. The pipeline for the data preparation stage is shown in Fig. 1.

Assuming a series of beat timing $\overrightarrow{\mathbf{T}} = (t_1, t_2, \ldots, t_n, t_{n+1})$where t_{n+1} represents the end of the last note. The tempo of the beat τ could be calculated as $\tau = \frac{60}{t_{i+1}-t_i}$. The raw vector of tempo $\mathbf{T} = (\tau_1, \tau_2, \ldots, \tau_n)$ is smoothed by the moving average with a window of three samples. The tempo curve is then truncated according to phrase information and standardised with a regulated mean of 1.

A phrasing mask in introduced as a preparation of automatic phrasing in the future. For the input smoothed and standardised tempo curves $\hat{\mathbf{T}}$, four prior beats and four posterior beats of a phrase are also included. A phrasing mask $\mathbf{M}$ is applied with the beats in a phrase labelled as "1" and the other beats as "0". The expressive timing within a phrase can be taken as the element-wise product between the standardised tempo curve and the mask, i.e. $\breve{\mathbf{T}} = \hat{\mathbf{T}} \odot \mathbf{M}$.

Classical data argumentation methods are then applied for better data availability and generalisation. The data augmentation methods used are:

- **Gaussian Noise**: Adds noise $\mathbf{n} \sim \mathcal{N}(0, \sigma)$, $\sigma \sim \text{Uniform}(0.05, 0.15)$, clipped to ensure non-negativity: $\breve{\mathbf{T}}' = \text{clip}(\breve{\mathbf{T}} + \mathbf{n}, 0, \max(\breve{\mathbf{T}}))$.
- **Scaling**: Multiplies by $s \sim \text{Uniform}(0.8, 1.2)$: $\breve{\mathbf{T}}' = s \cdot \breve{\mathbf{T}}$.
- **Local Perturbation**: Perturbs a random 10-step segment starting at $k \sim \text{Uniform}(0, T - 10)$ by $p \sim \text{Uniform}(-0.2, 0.2)$, clipped: $\breve{\mathbf{T}}'[k : k + 10] = \text{clip}(\breve{\mathbf{T}}[k : k + 10] + p, 0, \max(\breve{\mathbf{T}}))$.
- **Sinusoidal Modulation**: Adds $0.05 \cdot \sin(2\pi f t/T)$, $f \sim \text{Uniform}(0.5, 1.5)$, $t = [0, \ldots, T - 1]$, with clipping.

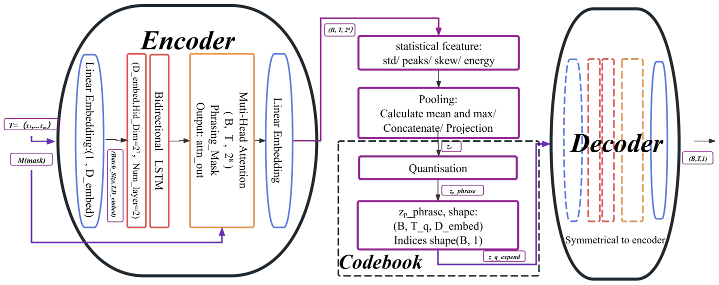


Fig. 2: The complete structure of the VQ-VAE model in the experiment.

3.2 VQ-VAE

The proposed VQ-VAE system has an asymmetric code structure as shown in Fig. 2. In the encoder, the argumented masked standardised tempo curve is padded with zeros for the identical lengths of inputs within the same batch.

Encoder The padded tempo curves undergo an initial linear transformation to increase their dimensionality before further processing. The proposed system employs a two-stage strategy: a Bi-directional LSTM (BLSTM) layer to extract local temporal features from the transformed input, followed by a multi-head attention layer to capture global temporal features. The time step for the BLSTM is set equal to the padded tempo curve length.

The subsequent multi-head attention layer processes the local temporal features extracted by the BLSTM, incorporating information from phrasing masks. The attention layer's time step is aligned with the phrase lengths. Essentially, the attention layer inverts the mask to assign higher attention weights to interior beats, prioritising expressive timing gestures while ignoring boundary beats. The interaction between queries and keys in the attention layer allows indirect incorporation of contextual information from all beats, as the input features retain the full sequence context. The value emphasises interior patterns while maintaining global context, serving as input to both boundary prediction and latent encoding.

With another layer of linear transform, the output of the attention layer, denoted as $\mathbf{A}$, is mapped to a continuous latent space:

$$\mathbf{z}_e = \mathbf{W}_f \mathbf{A} + \mathbf{b}_f, \tag{1}$$

where $\mathbf{W}_f \in \mathbb{R}^{E \times H}$, $\mathbf{b}_f \in \mathbb{R}^E$, and E is the codebook embedding dimension, producing $\mathbf{z}_e \in \mathbb{R}^{B \times T \times E}$. Each $\mathbf{z}_{e,b,t} \in \mathbb{R}^E$ represents a latent vector for time step t in batch b, encoding refined features that combine interior-focused attention and full-sequence context. $\mathbf{z}_e$ is used for computing the phrase-level representation and quantisation.

Latent Space and Sampling As the classical VAE, the encoder integrates statistical features and temporal information to compute a phrase-level latent vector $\mathbf{z}_{e,\text{phrase}} \in \mathbb{R}^E$. First, statistical features are computed from the masked sequence $\mathbf{x} \odot \mathbf{m}$, which isolates interior beats to capture phrase-specific rhythmic characteristics:

- Standard deviation: $\sigma_b = \sqrt{\frac{1}{\sum m_{b,t}} \sum_t m_{b,t}(x_{b,t} - \mu_b)^2}$, $\mu_b = \frac{\sum_t m_{b,t} x_{b,t}}{\sum m_{b,t}}$, measuring the variability of tempo curve.
- Peak count: $p_b = \frac{\sum_t \nVdash (x_{b,t} > 1 + 2\sigma_b) m_{b,t}}{\sum m_{b,t}}$, counting significant peaks on tempo curves.
- Skewness: $s_b = \frac{\sum_t m_{b,t}(x_{b,t} - 1)^3}{\sigma_b^3 \sum m_{b,t}}$, assessing skewness of tempo curves.
- Spectral energy: $e_b = \frac{1}{5} \sum_{k=1}^{5} |\text{FFT}(\mathbf{x}_b \odot \mathbf{m}_b)_k|$, capturing frequency-domain energy for tempo curves.

These form $\mathbf{s}_b = [\sigma_b, p_b, s_b, e_b] \in \mathbb{R}^4$ are embedded via:

$$\mathbf{s}_e = \mathbf{W}_s \mathbf{s}_b + \mathbf{b}_s, \tag{2}$$

where $\mathbf{W}_s \in \mathbb{R}^{E \times 4}$, $\mathbf{s}_e \in \mathbb{R}^E$.

Next, the masked-weighted mean and maximum of $\mathbf{z}_e$ are calculated to summarise the sequence while emphasising interior regions:

$$\mathbf{z}_{e,\text{mean}} = \frac{\sum_t \mathbf{z}_{e,b,t} m_{b,t}}{\sum m_{b,t}}, \quad \mathbf{z}_{e,\text{max}} = \max_t(\mathbf{z}_{e,b,t} \cdot m_{b,t}), \tag{3}$$

where $\mathbf{z}_{e,\text{mean}}, \mathbf{z}_{e,\text{max}} \in \mathbb{R}^E$ capture the average and peak latent features of interior beats, respectively, focusing on core musical content via the mask $\mathbf{m}$. These are concatenated with $\mathbf{s}_e$ to form $[\mathbf{z}_{e,\text{mean}}; \mathbf{z}_{e,\text{max}}; \mathbf{s}_e] \in \mathbb{R}^{3E}$, which is projected via:

$$\mathbf{z}_{e,\text{phrase}} = \mathbf{W}_p[\mathbf{z}_{e,\text{mean}}; \mathbf{z}_{e,\text{max}}; \mathbf{s}_e] + \mathbf{b}_p, \tag{4}$$

where $\mathbf{W}_p \in \mathbb{R}^{E\times 3E}$, $\mathbf{b}_p \in \mathbb{R}^E$. This integrates temporal (from LSTM/attention) and statistical information into a single vector per phrase, connecting the sequence-level $\mathbf{z}_e$ to a compact representation for quantisation.

Quantisation in Latent Space The phrase-level representation $\mathbf{z}_{e,\text{phrase}}$ connects the encoder to the quantisation layer, where it is mapped to a discrete codebook vector:

$$\mathbf{z}_q = \mathbf{C}\big[\arg\min_k \|\mathbf{z}_{e,\text{phrase}} - \mathbf{c}_k\|_2^2\big], \tag{5}$$

using a codebook $\mathbf{C} \in \mathbb{R}^{K\times E}$, where K is the number of embeddings. Indices are sampled via softmax with temperature $\tau = 0.5$:

$$p(k) = \frac{\exp(-\|\mathbf{z}_{e,\text{phrase}} - \mathbf{c}_k\|_2^2/\tau)}{\sum_{k'} \exp(-\|\mathbf{z}_{e,\text{phrase}} - \mathbf{c}_{k'}\|_2^2/\tau)}. \tag{6}$$

The quantised vector is expanded to $\mathbf{z}_q \in \mathbb{R}^{B\times T\times E}$, linking the compact representation from the encoder to the decoder for reconstruction.

Decoder The decoder mirrors the encoder in a symmetric structure, reversing the process to reconstruct tempo curves from quantised latent representations, using a multi-head attention layer followed by a Bi-directional LSTM layer, and concluding with a linear transform to output the final sequence.

3.3 Clustering

The clustering method for tempo curve analysis employs a Vector Quantised Variational Auto-Encoder (VQ-VAE) with a discrete codebook. The codebook is initialised using mini-batch K-Means on tempo curve z_e, sampled up to 200,000 points with a batch size of 3,072 and 50 initialisations, minimising:

$$J = \sum_{i=1}^{n} \min_{\mu_j \in C} \sum_{k=1}^{d} (z_{i,k} - \mu_{j,k})^2. \tag{7}$$

Codebook vectors w_j are initialised uniformly in $[-\frac{1}{k}, \frac{1}{k}]$, where k is the number of embeddings, and standardised post-training to unit length, ensuring:

$$\sum_{i=1}^{d} w_{j,i}^2 = 1, \tag{8}$$

achieved via $w_j \leftarrow w_j / \sqrt{\sum_{i=1}^{d} w_{j,i}^2}$, projecting vectors onto the unit hypersphere.

Quantisation assigns z_e to codebook entries using squared Euclidean distance:

$$d(z_e, w_j) = \sum_{i=1}^{d} (z_{e,i} - w_{j,i})^2, \tag{9}$$

with a softmax probability (τ: 1.0 to 0.5 over epochs):

$$p(j \mid z_e) = \frac{\exp(-d(z_e, w_j)/\tau)}{\sum_{l=1}^{k} \exp(-d(z_e, w_l)/\tau)}. \tag{10}$$

Assignments are processed in blocks of 1,000. Clusters are merged if $\sum_{k=1}^{d}(\mathbf{w}_{i,k} - \mathbf{w}_{j,k})^2 <$ threshold (e.g., 0.0016), where $\mathbf{w}_i, \mathbf{w}_j \in \mathbb{R}^d$ represents codebook vectors, and D is the embedding dimension.

4 Experiments and Results

4.1 Datasets

The datasets used in this paper is a combination of two datasets. The public Mazurka datasets and the private Islamey dataset. For Mazurka dataset, there are five pieces of Chopin Mazurka with multiple performances by different pianists: Op. 17/4 (13 phrases, 63 performances), Op. 24/2 (30 phrases, 64 performances), Op. 30/2 (8 phrases, 34 performances), Op. 63/3 (13 phrases, 95 performances) and Op. 68/3 (10 phrases, 51 performances). The private Islamey dataset contains 25 performances where each performance has 40 phrases. In total, there are 5756 tempo curves for intra-phrase expressive timing.

4.2 Training of VQ-VAE

The training process is performed with a GPU of RTX 4060ti(8G) with a implementation with PyTorch with the hyper-parameter shown in Table 1. The VQ-VAE is trained with a composite loss with all items using mean squared error:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{vq}} + \beta\mathcal{L}_{\text{commit}}, \tag{11}$$

where:

- $\mathcal{L}_{\text{recon}} = \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2$, is in mean squared error, ensuring $\hat{\mathbf{x}}$ matches $\mathbf{x}$.
- $\mathcal{L}_{\text{vq}} = \|\text{sg}[\mathbf{z}_e] - \mathbf{z}_q\|_2^2$, aligning $\mathbf{z}_e$ with $\mathbf{z}_q$.

- $\mathcal{L}_{\text{commit}} = \|\mathbf{z}_e - \text{sg}[\mathbf{z}_q]\|_2^2$, committing $\mathbf{z}_e$ to the codebook.

With the use of the AdamW optimiser with a cosine annealing scheduler, the training process stops when both training and validation losses do not decrease by a minimum threshold for five consecutive epochs.

Table 1: Hyper-parameters of the VQ-VAE model that is used in the training process. The codebook used is specified in the context separately.

Parameter Name	**Value**	**Parameter Name**	**Value**
Input Dimension	64	Weight Decay	0.01
Hidden Dimension	256	Commitment Loss Weight	2
Attention Heads	4	Codebook Diversity Weight	0.15
Batch Size	8	Initial Quantisation Temperature	1.0
Max Epochs	40	Final Quantisation Temperature	0.5
Learning Rate	0.0005	Quantisation Block Size	1000

4.3 Number of Clusters for Expressive Timing

The resulting number of clusters for expressive timing in the proposed analysis will be presented first. There are three factors impact the resulting number of clusters: the size of codebook (i.e. the number of codes in a codebook), the length of code (i.e. the dimensional of embedded coding) and the threshold that merges similar vectors.

According Gratton et al. [21], the Just Noticeable Difference (JND) in the circumstances of expressive timing could be around 12%. For a consistent sequence of stimuli, the JND may be significantly improved to 3%. Considering the expressive timing within a phrase could be steady (but no consistent) under special circumstances, the JND of expressive timing in the proposed experiment is taken as 4%, which corresponds to a threshold of 0.0016 due to the normalised codes.

With the given threshold, the range of code length is decided. Given the fact that the phrase lengths in the dataset are usually no more than 24 beats, the candidate number of dimensions in the codebook is proposed to 32, 64, 128, 256 and 512, where the lowest dimension should be able to keep all information of tempo curves without loss.

Moreover, the number of clusters for intra-phrase expressive timing is usually less than 10 in a piece of music [2]. As there are six pieces of music, the possible number of clusters for the whole dataset ranges from around 10 (assuming the same clusters of expressive timing across pieces) to 60 (if no clusters of expressive timing is shared between pieces). As a result, the size of the codebook in VQ-VAE is set to 8, 16, 32, 64 and 128, which set the maximum number of clusters in expressive timing from marginally fit to more than adequacy.

Table 2: The number of clusters resulted by VQ-VAE with different settings of tables. D_{dim} refers to the dimension of code (or code length) and D_{size} refers the number of codes in codebook (i.e. codebook size).

	$D_{\text{dim}} = 32$	$D_{\text{dim}} = 64$	$D_{\text{dim}} = 128$	$D_{\text{dim}} = 256$	$D_{\text{dim}} = 512$
$D_{\text{size}} = 8$	1	1	1	1	1
$D_{\text{size}} = 16$	1	1	2	3	7
$D_{\text{size}} = 32$	1	1	3	12	20
$D_{\text{size}} = 64$	1	2	2	61	26
$D_{\text{size}} = 128$	4	4	125	128	127

As shown in Table 2, a larger size of codebook and a larger dimension of code lead to more clusters in general. However, there are few exceptions: with $D_{\text{size}} = 64$, $D_{\text{dim}} = 256$ leads to 61 clusters whereas $D_{\text{dim}} = 512$ leads to only 26 clusters. There are only a few minor cases (resulting 1 more or 1 less cluster).

To select the number of clusters, the model selection test [2] is performed as the baseline. In short, Assuming $f(t) = \sum_{i=0}^{4} a_i x^i$, the regressed polynomial coefficients $\mathbf{A} = (a_0, a_1, a_2, a_3, a_4)$ for $\breve{\mathbf{T}}$ can be fitted to a Gaussian Mixture Model (GMM) for clustering.

The whole dataset used in the experiment is regressed by a polynomial function of a fourth order. A Gaussian Mixture Model (GMM) with the value of k set between 2 to 128 is trained. AIC (Akaike Information Criterion) measures the performance of resulting GMMs as demonstrated by Figure 3. According to the model selection test, GMM with 12 Gaussian components outperforms other models, which leads to 12 clusters of expressive timing. Given the experiment presented in Table 2, we choose a codebook with $D_{\text{dim}} = 256$ and $D_{\text{size}} = 32$.

4.4 Evaluation with Cluster Distribution

The validation of expressive timing clustering involves comparing the cluster distributions from the VQ-VAE model against those from a classical Gaussian Mixture Model (GMM) [2]. The difference between the two distributions, resulted from GMMs (Q) and VQ-VAE systems (P) is quantified using Jensen-Shannon (JS) Divergence. JS Divergence is a symmetrised measure based on Kullback-Leibler (KL) Divergence. The KL Divergence is defined as: $\text{KL}(P||Q) = \sum_{x \in \chi} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$ where x represents a specific cluster. The resulting JS Divergence is calculated as: $\text{JS}(P||Q) = \frac{1}{2}(\text{KL}(P||Q) + \text{KL}(Q||P))$.

The clustering results are considered valid if the divergence between the VQ-VAE and GMM distributions is comparable to the natural variability found within GMMs run with different initial conditions. Ten VQ-VAE systems, using slightly varied thresholds for the same number of clusters (set between 0.0016 to 0.0020, 4% to 4.5% differences on average), are compared against ten GMMs with different initialisations.

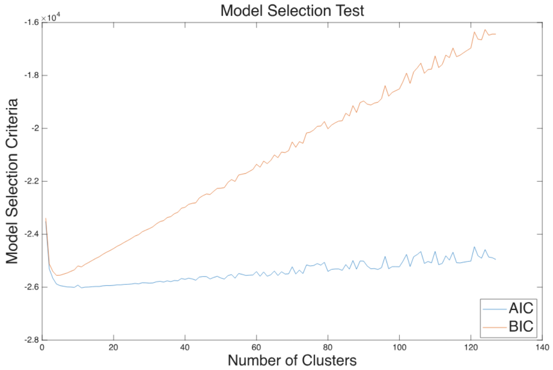
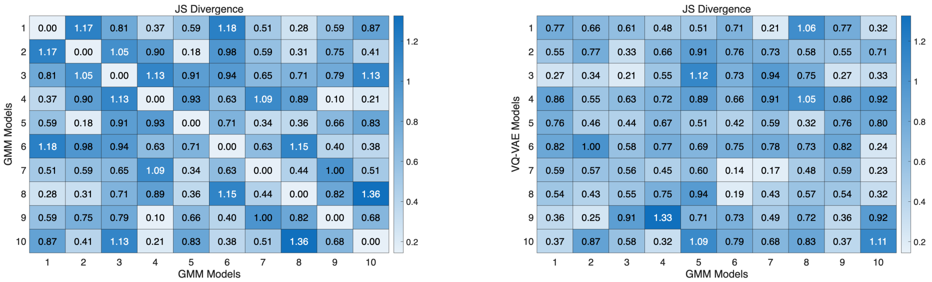


Fig. 3: Model selection test for all data with regressed polynomial coefficients. A more negative value indicates better performance. The best AIC is resulted by 12 clusters.



	1	2	3	4	5	6	7	8	9	10
1	0.00	1.17	0.81	0.37	0.59	1.18	0.51	0.28	0.59	0.87
2	1.17	0.00	1.05	0.90	0.18	0.98	0.59	0.31	0.75	0.41
3	0.81	1.05	0.00	1.13	0.91	0.94	0.65	0.71	0.79	1.13
4	0.37	0.90	1.13	0.00	0.93	0.63	1.09	0.89	0.10	0.21
5	0.59	0.18	0.91	0.93	0.00	0.71	0.34	0.36	0.66	0.83
6	1.18	0.98	0.94	0.63	0.71	0.00	0.63	1.15	0.40	0.38
7	0.51	0.59	0.65	1.09	0.34	0.63	0.00	0.44	1.00	0.51
8	0.28	0.31	0.71	0.89	0.36	1.15	0.44	0.00	0.82	1.36
9	0.59	0.75	0.79	0.10	0.66	0.40	1.00	0.82	0.00	0.68
10	0.87	0.41	1.13	0.21	0.83	0.38	0.51	1.36	0.68	0.00

(a) JS Divergence between each pair of GMM results

	1	2	3	4	5	6	7	8	9	10
1	0.77	0.66	0.61	0.48	0.51	0.71	0.21	1.06	0.77	0.32
2	0.55	0.77	0.33	0.66	0.91	0.76	0.73	0.58	0.55	0.71
3	0.27	0.34	0.21	0.55	1.12	0.73	0.94	0.75	0.27	0.33
4	0.86	0.55	0.63	0.72	0.89	0.66	0.91	1.05	0.86	0.92
5	0.76	0.46	0.44	0.67	0.51	0.42	0.59	0.32	0.76	0.80
6	0.82	1.00	0.58	0.77	0.69	0.75	0.78	0.73	0.82	0.24
7	0.59	0.57	0.56	0.45	0.60	0.14	0.17	0.48	0.59	0.23
8	0.54	0.43	0.55	0.75	0.94	0.19	0.43	0.57	0.54	0.32
9	0.36	0.25	0.91	1.33	0.71	0.73	0.49	0.72	0.36	0.92
10	0.37	0.87	0.58	0.32	1.09	0.79	0.68	0.83	0.37	1.11

(b) (b) JS Divergence between VQ-VAE and GMM results

Fig. 4: The JS divergence of cluster distribution between GMM method and VQ-VAE system. The index of GMM models at the X-axis are the same model.

The average difference between resulting GMMs and VQ-VAEs are shown in Fig. 4. The average divergence between GMMs caused by different initialisations was 0.71 ± 0.32. In contrast, the divergence between the VQ-VAE system and the GMMs was 0.62 ± 0.24. Therefore, the VQ-VAE system produces a similar distributions of clusters with a GMM model.

The resulting expressive timing clusters are considered valid as the difference between the proposed system and a GMM is both more stable and similar to the natural variability within GMMs caused by different starting conditions.

5 Future Work

The current VQ-VAE system effectively clusters phrase-level expressive timing categories, demonstrating its capability to handle variable-length sequences.

However, its application is limited to statistical clustering without deeper insights into musical expression, and its evaluation is confined to classical piano repertoire from specific datasets.

To enhance the model's musical relevance, future work will extend it by integrating a supervised learning task for automatic phrasing alongside the existing unsupervised clustering framework. This will involve a multi-task learning approach where the encoder learns to predict phrasing boundaries using expert-annotated labels of musical scores. The trained model will automatically segment scores into phrases for specific musical styles, such as classical or other genres, while clustering expressive timing within those phrases. By linking clusters to interpretable phrasing structures, this approach will provide valuable insights into performance expression. Additionally, the codes in codebook need to be interpreted as the common gestures of expressive timing to enable further expressiveness analysis although the clustering process is demonstrated valid.

6 Conclusion

This paper proposes a VQ-VAE system that clusters expressive timing within a phrase. The method takes the codebook in VQ-VAE as a means of clustering method, which enable the adaptivity of cluster numbers and the potential of adapting different phrasing strategies. As the result shows that a higher maximum number clusters allowed leads to a solution with more clusters. The length of code is set to be 64 given the dataset combining Mazurka dataset and the private Islamey dataset, whose phrase lengths are no more than 24 beats in most cases. The method is validated via the comparison of the classical GMM based clustering method with regressed polynomial coefficients for intra-phrase tempo curves. With a similar number of clusters resulted with the model selection test with GMMs, the cluster distribution between the proposed system and the traditional GMM system is smaller than the difference in cluster distribution by different GMMs.

References

1. Demos, A.P., Lisboa, T., Begosh, K.T., Logan, T., Chaffin, R.: A longitudinal study of the development of expressive timing. Psychology of Music **48**, 50–66 (2020)
2. Li, S., Dixon, S., Plumbley, M.D.: Clustering expressive timing with regressed polynomial coefficients demonstrated by a model selection test. In: Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR), pp. 646–652 (2017)
3. Van Den Oord, A., Vinyals, O., Kavukcuoglu, K.: Neural discrete representation learning. In: Advances in Neural Information Processing Systems (2017)
4. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
5. Nielsen, F.: On the Jensen–Shannon Symmetrization of Distances Relying on Abstract Means. Entropy **21** (2019). https://doi.org/10.3390/e21050485

6. Clarke, E.F.: Expression in performance: generativity, perception and semiosis. The practice of performance: Studies in musical interpretation, pp. 21–54 (1995)
7. Xiao, Z., Chen, X., Zhou, L.: Music performance style transfer for learning expressive musical performance. Signal, Image and Video Processing **18**, 889–898 (2024)
8. Kim, H., Choi, S., Nam, J.: Expressive acoustic guitar sound synthesis with an instrument-specific input representation and diffusion outpainting. In: ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7620–7624 (2024)
9. Dai, S., Liu, M.-Y., Valle, R., Gururani, S.: Expressivesinger: Multilingual and multi-style score-based singing voice synthesis with expressive performance control. In: Proceedings of the 32nd ACM International Conference on Multimedia, pp. 3229–3238 (2024)
10. Cancino-Chacón, C.E., Grachten, M., Goebl, W., Widmer, G.: Computational Models of Expressive Music Performance: A Comprehensive and Critical Review. Frontiers in Digital Humanities **5**, 25 (2018). https://doi.org/10.3389/fdigh.2018.00025
11. Repp, B.H.: A microcosm of musical expression. I. Quantitative analysis of pianists' timing in the initial measures of Chopin's Etude in E major. The Journal of the Acoustical Society of America **104**, 1085–1100 (1998)
12. Spiro, N., Gold, N., Rink, J.: The form of performance: analyzing pattern distribution in select recordings of Chopin's Mazurka Op. 24 No. 2. Musicae Scientiae **14**, 23–55 (2010)
13. Li, S., Black, D.A., Plumbley, M.D.: The Clustering of Expressive Timing Within a Phrase in Classical Piano Performances by Gaussian Mixture Models. In: International Symposium on Computer Music Multidisciplinary Research (CMMR), pp. 322–345 (2015)
14. Shi, Z.: Computational analysis and modeling of expressive timing in Chopin's Mazurkas. In: ISMIR, pp. 650–656 (2021)
15. Alqahtani, A., Ali, M., Xie, X., Jones, M.W.: Deep time-series clustering: A review. Electronics **10**, 3001 (2021)
16. Mukherjee, S., Asnani, H., Lin, E., Kannan, S.: Clustergan: Latent space clustering in generative adversarial networks. In: Proceedings of the AAAI conference on artificial intelligence, pp. 4610–4617 (2019)
17. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: International conference on machine learning, pp. 478–487 (2016)
18. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
19. Dhariwal, P., Jun, H., Payne, C., Kim, J.W., Radford, A., Sutskever, I.: Jukebox: A generative model for music. arXiv preprint arXiv:2005.00341 (2020)
20. Lee, D., Malacarne, S., Aune, E.: Vector quantized time series generation with a bidirectional prior model. arXiv preprint arXiv:2303.04743 (2023)
21. Gratton, I., Brandimonte, M.A., Bruno, N.: Absolute memory for tempo in musicians and non-musicians. PloS one **11**, e163558 (2016)