MusiCode: Discrete Representation of Symbolic Music for Data Mining

Lekai Qian[1,*], Tingran Meng[1,*], Chenyuan Hong[1,*], Haoyu Gu[1,*], Shidang Xu[2,†]
[1]School of Future Technology, South China University of Technology
[2]School of Biomedical Sciences and Engineering, South China University of Technology
{202364870191, 202364870171, 202364870042, 202364820071}@mail.scut.edu.cn
xusd@scut.edu.cn
*Equal contribution. †Corresponding author.

Abstract

Existing symbolic music processing methods predominantly rely on token-based representations, which generate extremely long sequences and struggle to capture global structural patterns. To address these challenges, we propose MusiCode, a framework combining representation learning, data mining, and vector quantization for music analysis. Our framework consists of three components: (1) a Transformer-based autoencoder learning compact 768-dimensional embeddings through self-supervised reconstruction, (2) systematic data mining techniques including PCA, t-SNE, UMAP, and K-Means clustering to discover latent patterns and visualize musical relationships, and (3) a theoretical framework for Residual Vector Quantization (RVQ) enabling hierarchical discrete encoding. We implement and validate the first two components through experiments on 10,000 music segments. Our clustering analysis reveals meaningful patterns in the embedding space, with partial separation between high and low voice parts, demonstrating that the Transformer autoencoder captures voice-related structural information. While the RVQ framework is theoretically designed for future implementation, our results validate the effectiveness of combining learned representations with data mining techniques for music analysis, providing a foundation for discrete encoding and compression tasks.

1 Introduction

The rapid growth of digital music libraries and streaming platforms has generated unprecedented volumes of symbolic music data, creating both opportunities and challenges for music information retrieval and analysis. Understanding the underlying patterns, structures, and relationships within these vast musical collections requires sophisticated computational approaches that can capture both local musical features and global structural information. While significant progress has been made in symbolic music generation and analysis, fundamental questions remain about how to effectively represent and compress musical information in ways that preserve semantic content while enabling efficient computation and analysis.

1.1 Motivation and Background

Traditional approaches to symbolic music processing have predominantly relied on token-based representations, where music is encoded as sequences of discrete events—such as note onsets, pitches, durations, and velocities. These methods have demonstrated impressive capabilities in music generation tasks, successfully producing locally coherent musical sequences through autoregressive modeling and transformer architectures. However, this event-level granularity, while temporally precise, introduces several fundamental limitations that constrain both the efficiency and expressiveness of musical representations.

First, token-based approaches generate extremely long sequences even for short musical passages. A simple four-measure phrase may require hundreds of discrete tokens, leading to computational bottlenecks in both training and inference. Second, and more critically, these representations struggle to capture global structural information and high-level musical patterns. By operating at the level of individual events, token-based models must implicitly learn harmonic progressions, rhythmic patterns, and thematic relationships through statistical regularities in vast training datasets, without explicit encoding of these musically meaningful structures. This results in models that can generate locally plausible continuations but often fail to maintain long-term coherence or meaningful global structure.

Recent work in variational autoencoders has demonstrated that learning continuous latent representations can enable better capture of musi-

cal structure. However, these approaches typically operate downstream of pre-defined tokenization schemes, inheriting their limitations. Similarly, while diffusion models have shown promise for symbolic music generation, they remain constrained by the representations on which they operate. These observations motivate our exploration of alternative approaches that combine learned representations with systematic data mining techniques to discover and analyze musical patterns.

1.2 Our Approach: Representation Learning and Data Mining

In this work, we propose a comprehensive framework that addresses these limitations through the integration of representation learning, data mining techniques, and vector quantization. Our approach, **MusiCode**, consists of three complementary components designed to enable effective music analysis and compression. **In this work, we focus particularly on the first two components—representation learning and data mining—demonstrating their effectiveness through comprehensive experiments on 83,362 music token embeddings.** We also present the theoretical framework for the third component, residual vector quantization, providing a foundation for future implementation and integration.

Representation Learning: We employ a Transformer-based autoencoder architecture to learn compact, continuous representations of symbolic music. Rather than operating directly on lengthy token sequences, our model learns to compress musical segments into fixed-dimensional 768-dimensional embeddings through self-supervised reconstruction. This approach, inspired by recent advances in music representation learning such as MusicVAE (Roberts et al., 2018), enables the model to capture essential musical information in a significantly more compact form. By training the autoencoder to reconstruct input sequences from these compressed representations, we ensure that the learned embeddings preserve musically meaningful information.

Data Mining and Visualization: We systematically apply data mining techniques to analyze the learned representation space and discover latent musical patterns. We first employ UMAP (McInnes et al., 2018) for dimensionality reduction, which preserves both local and global structure more effectively than traditional methods like PCA or t-SNE (van der Maaten and Hinton, 2008). For clustering, we use HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) (Campello et al., 2013), a density-based clustering algorithm that offers several advantages over traditional K-Means: (1) it automatically determines the optimal number of clusters without requiring pre-specification, (2) it can identify clusters of arbitrary shapes rather than assuming spherical distributions, (3) it robustly handles noise by identifying outlier points, and (4) it provides a hierarchical view of cluster structure. Through systematic parameter search over the UMAP-HDBSCAN pipeline, we discover natural groupings within the embedding space, potentially corresponding to voice types, musical styles, or structural patterns.

Hierarchical Vector Quantization (Theoretical Framework): To bridge continuous feature representations and discrete symbolic manipulation, we design a theoretical framework based on Residual Vector Quantization (RVQ) (Zeghidour et al., 2021; van den Oord et al., 2017). RVQ uses multiple quantization layers in a hierarchical manner, where each layer progressively refines the representation by quantizing residual errors from previous layers. This approach offers several critical advantages: (1) it achieves high-quality discrete encoding while maintaining compact codebook sizes, (2) it naturally captures hierarchical structure, with early layers encoding coarse musical characteristics and later layers capturing fine-grained details, and (3) it enables flexible rate-quality tradeoffs by adjusting the number of quantization layers. While we present the complete theoretical framework for RVQ in this work, its implementation and empirical evaluation are reserved for future research, building upon the learned representations validated in our current experiments.

1.3 Contributions

The main contributions of this work are:

- **Transformer-Based Music Representation Learning:** We implement and evaluate a Transformer autoencoder that learns compact 768-dimensional embeddings from token sequences through self-supervised reconstruction, demonstrating effective compression of musical information for subsequent analysis.

- **Systematic Data Mining Analysis:** We demonstrate how modern data mining techniques—UMAP dimensionality reduction

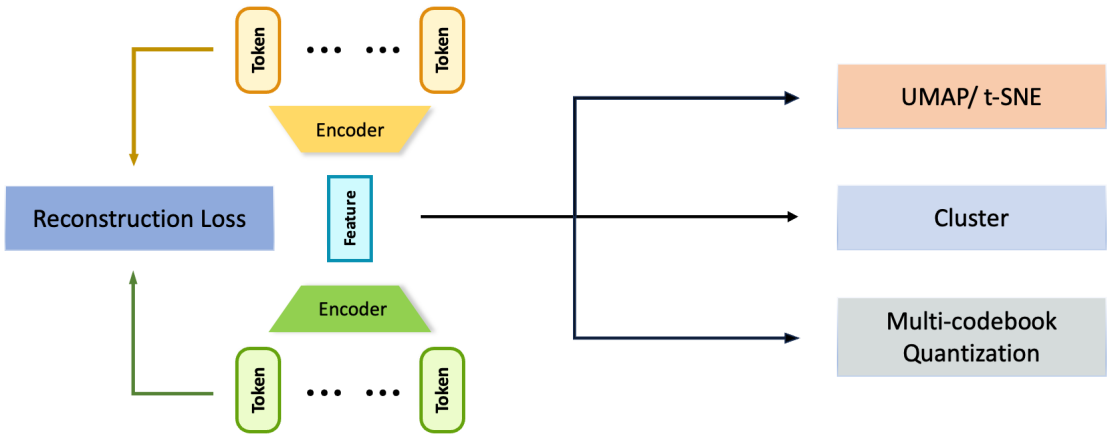

Figure 1: Overview of the MusiCode framework. The Transformer-based autoencoder (Component 1) compresses token sequences into 768-dimensional embeddings through self-supervised reconstruction. The learned embeddings are then analyzed through parallel data mining techniques (Component 2): UMAP dimensionality reduction and HDBSCAN clustering. Component 3 (shown with dashed border) provides a theoretical framework for residual vector quantization, reserved for future implementation.

and HDBSCAN density-based clustering—can be systematically applied to learned music representations to discover latent patterns and identify natural groupings. Through comprehensive parameter search over 360 configurations targeting approximately 512 clusters, our experiments on 83,362 music token embeddings reveal meaningful structure in the embedding space, with partial separation between high and low voice parts.

- **Theoretical Framework for Hierarchical Quantization:** We present a complete theoretical framework for Residual Vector Quantization applied to music feature spaces, providing the mathematical foundation and architectural design for future implementation of hierarchical discrete encoding.
- **Comprehensive Empirical Analysis:** We provide extensive clustering analysis and visualization demonstrating that learned representations capture meaningful musical structures. Our results validate the effectiveness of combining representation learning with data mining techniques for music analysis, providing a solid foundation for subsequent discrete encoding tasks.

Our approach demonstrates that learned representations, analyzed through data mining techniques, offer a promising direction for music analysis and understanding. By operating in a compact embedding space rather than lengthy token sequences, we enable more efficient analysis while capturing high-level musical patterns. The theoretical framework for hierarchical quantization provides a clear path for future work to extend these representations into discrete codes suitable for compression and generation tasks.

1.4 Organization

The remainder of this paper is organized as follows. Section 2 reviews related work in symbolic music representation, data mining in music analysis, and vector quantization techniques. Section 3 describes our representation learning architecture, data mining techniques, and the theoretical framework for residual vector quantization. Section 4 presents our experimental setup, datasets, and implementation details. Section 5 analyzes our results through visualizations of embedding spaces, clustering quality metrics, and pattern discovery. Section 6 discusses the implications of our findings, current limitations, and future directions. Section 7 concludes the paper.

2 Related Work

2.1 Data Mining in Music Analysis

Data mining techniques have been extensively applied to music information retrieval and analysis, offering powerful tools for discovering patterns and structures in large musical datasets (Tzanetakis and Cook, 2002; Li et al., 2003).

Clustering Methods: Clustering algorithms are fundamental for discovering natural groupings in

music data. K-Means clustering has been widely used for music genre classification (Tzanetakis and Cook, 2002) and automatic playlist generation. However, K-Means requires pre-specifying the number of clusters and assumes spherical cluster shapes, which may not suit complex musical data. Density-based methods like DBSCAN and its hierarchical variant HDBSCAN (Campello et al., 2013) offer advantages by automatically determining cluster numbers and identifying arbitrarily-shaped clusters while robustly handling noise. HDBSCAN constructs a hierarchy of clusters based on density estimates, extracting stable clusters across multiple scales. Hierarchical clustering and Gaussian Mixture Models (GMM) have also been applied to capture more complex cluster structures in music feature spaces (Rodà et al., 2014).

Dimensionality Reduction: High-dimensional music feature spaces pose significant challenges for visualization and analysis. Principal Component Analysis (PCA) has been a classical approach for reducing feature dimensionality while preserving maximum variance (Aucouturier et al., 2007). More recently, non-linear dimensionality reduction techniques have gained prominence. t-Distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten and Hinton, 2008) has become particularly popular for visualizing music datasets, as it excels at preserving local structure and revealing cluster patterns. Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018) offers similar visualization quality with superior computational efficiency and better preservation of global structure, making it suitable for large-scale music analysis tasks.

Pattern Recognition: Sequential pattern mining and association rule learning have been applied to discover common progressions in music, such as chord sequences and melodic motifs (Conklin, 2003). These techniques enable the extraction of recurring patterns that characterize different musical styles and genres.

Our work leverages these data mining techniques in a unified framework. We employ UMAP for dimensionality reduction due to its superior balance of local and global structure preservation and computational efficiency. For clustering, we adopt HDBSCAN rather than K-Means, as it automatically determines the optimal number of clusters and handles the complex, non-spherical cluster shapes common in learned music representations. By integrating these modern techniques with vector quantization methods, we create a comprehensive pipeline for music analysis.

2.2 Symbolic Music Representations and Feature Learning

The representation of music fundamentally determines what patterns and structures can be captured by computational models. Traditional approaches have evolved from raw MIDI event sequences (Oore et al., 2018) to more structured representations that better capture musical semantics.

Token-based Representations: Modern symbolic music processing heavily relies on tokenization schemes that convert MIDI data into discrete tokens. The REMI (REvamped MIDI-derived events) representation (Huang and Yang, 2020) introduced explicit metrical tokens (Bar, Position) to provide rhythmic context. Compound Word representations (Hsiao et al., 2021) further group related tokens (pitch, duration, velocity) into unified compounds, reducing sequence length while capturing co-occurrence relationships. These tokenization methods demonstrate that appropriate discrete representations can significantly improve model performance, motivating our exploration of vector quantization for music feature encoding.

Pre-trained Music Representations: Large-scale pre-training has proven effective for learning contextualized music representations. MusicBERT (Zeng et al., 2021) and MidiBERT-Piano (Chou et al., 2021) apply BERT-style masked language modeling to symbolic music, learning representations that excel at downstream tasks such as composer classification and melody extraction. However, these models operate on pre-defined tokenization schemes and do not address the challenge of learning compressed, hierarchical feature representations directly from raw musical content.

Latent Representations with VAEs: Variational Autoencoders offer a complementary approach by learning continuous latent representations. MusicVAE (Roberts et al., 2018) introduced hierarchical decoder architectures that learn structured latent spaces, enabling semantically meaningful interpolation between musical pieces (Figure 2). The hierarchical structure, where a "conductor" RNN generates subsequence embeddings and separate "track" RNNs decode each subsequence, forces the model to utilize its latent code effectively. Extensions include style-aware representations (Valenti et al., 2020) and emotion-controlled generation (Grekow and Dimitrova-Grekow, 2021).

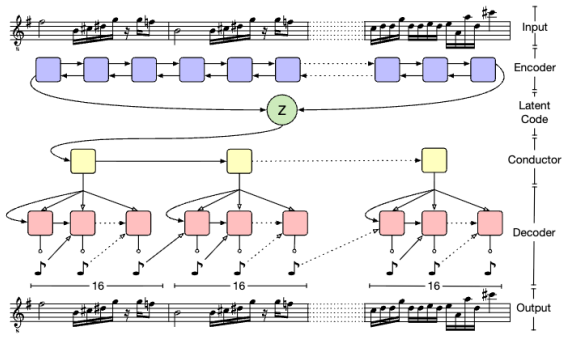

Figure 2: MusicVAE hierarchical architecture. The encoder compresses music sequences into latent code z, decoded hierarchically through a conductor RNN and multiple decoder RNNs.

While these methods demonstrate the value of learned continuous representations, they do not address the discrete quantization problem that enables efficient compression and symbolic manipulation—a gap our work addresses through residual vector quantization.

Hierarchical and Structured Representations: PianoTree VAE (Wang et al., 2020) explicitly models tree-structured note groupings (chords, arpeggios) to better capture polyphonic structure. Graph-based approaches (Jeong et al., 2019; Rodriguez-Lopez and Volk, 2018) represent musical relationships through explicit graph structures. These works highlight the importance of capturing hierarchical and relational information in music representation. Our approach achieves similar goals—preserving both local details and global structure—through multi-layer residual vector quantization operating on continuous feature representations.

2.3 Vector Quantization in Music

Vector quantization techniques have recently gained attention for creating discrete representations while maintaining high reconstruction quality. In the audio domain, SoundStream (Zeghidour et al., 2021) demonstrated that residual vector quantization (RVQ) can efficiently compress speech and music at various bitrates by using multiple quantization layers in a hierarchical manner (Figure 3).

As shown in Figure 3, the RVQ framework operates by progressively refining the quantization: each layer quantizes the residual error from previous layers, enabling flexible rate-quality tradeoffs. The encoder produces continuous latent representations, which are then quantized through multiple RVQ layers ($Q_1, Q_2, Q_3, Q_4, \ldots$), with each layer reducing the reconstruction error. This hierarchical structure naturally captures information at multiple levels of abstraction—early layers encode coarse structure while later layers capture fine-grained details.

Our work extends these ideas to symbolic music analysis by applying RVQ to learned feature representations rather than raw audio. This allows us to discover discrete codes that capture high-level musical patterns (harmonic progressions, rhythmic motifs) while maintaining computational efficiency. By combining feature extraction with hierarchical quantization, we bridge continuous feature learning and discrete symbolic manipulation.

2.4 Diffusion Models for Music

Recent work has explored diffusion models for symbolic music generation (Mittal et al., 2021; Plasser et al., 2023). These approaches operate in either continuous latent spaces (Mittal et al., 2021) or directly on discrete representations (Plasser et al., 2023; Zhang et al., 2024a). Hierarchical cascaded diffusion (Zhang et al., 2024b) and rule-guided diffusion (Huang et al., 2024) further enhance controllability and structural coherence.

While our work focuses on representation learning rather than generation, the discrete codes produced by our RVQ framework are compatible with these diffusion-based approaches. The hierarchical nature of RVQ—where early layers capture coarse structure and later layers encode fine details—aligns well with hierarchical diffusion models that operate at multiple abstraction levels.

3 Methodology

3.1 Overview and Framework Design

Our framework, MusiCode, consists of three complementary components designed to enable comprehensive music analysis and representation. Figure 1 illustrates the overall architecture.

Component 1: Representation Learning employs a Transformer-based autoencoder to learn compact 768-dimensional embeddings from token sequences through self-supervised reconstruction. The encoder processes input token sequences into fixed-dimensional continuous representations, which the decoder reconstructs back to the original sequence. The reconstruction loss ensures that embeddings capture essential musical information.

Component 2: Data Mining and Visualization applies modern data mining techniques

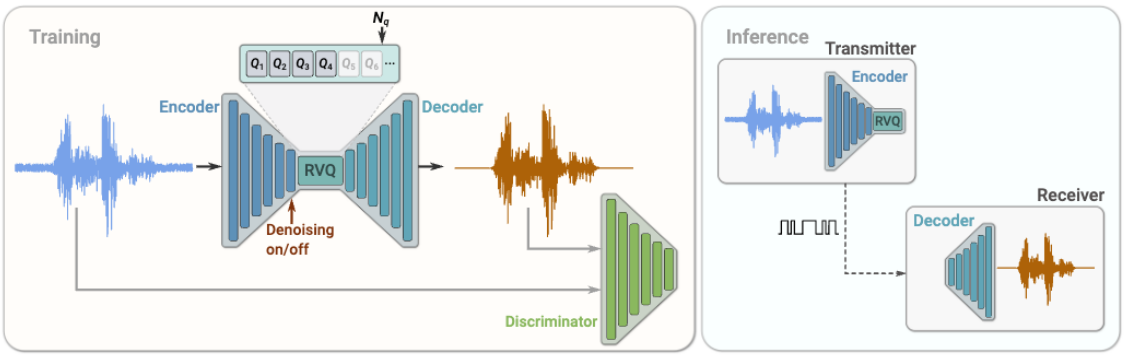

Figure 3: SoundStream architecture with Residual Vector Quantization (RVQ). Left: Training phase with encoder, multi-layer RVQ, decoder, and discriminator. Right: Inference phase showing encoder-RVQ compression at transmitter and RVQ-decoder reconstruction at receiver. The hierarchical RVQ structure enables flexible bitrate control and progressive refinement.

to analyze the learned embedding space. The 768-dimensional embeddings undergo a two-stage pipeline: (1) UMAP dimensionality reduction to a more tractable intermediate dimension (50-100D), preserving both local and global manifold structure, and (2) HDBSCAN density-based clustering to automatically discover natural groupings without pre-specifying cluster numbers. Through systematic parameter search, we optimize this pipeline to identify meaningful musical categories.

Component 3: Residual Vector Quantization provides a theoretical framework for converting continuous embeddings into hierarchical discrete codes. This enables efficient compression and symbolic manipulation while preserving musical information at multiple levels of abstraction.

In this work, we implement and evaluate Components 1 and 2 (Sections 3.2 through 3.5), demonstrating their effectiveness on 83,362 music token embeddings. Component 3 is presented as a theoretical framework (Section 3.6) with complete mathematical formulation, providing a foundation for future implementation.

3.2 Representation Learning with Transformer Autoencoder

We employ a Transformer-based autoencoder architecture to learn compact continuous representations of symbolic music. The model takes token sequences as input and learns to compress them into fixed-dimensional embeddings through self-supervised reconstruction training.

3.2.1 Architecture

Our autoencoder consists of an encoder E and a decoder D. Given an input token sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ where T is the sequence length, the encoder processes it through multiple Transformer layers to produce a continuous embedding:

$$\mathbf{z} = E(\mathbf{x}) \in \mathbb{R}^d \tag{1}$$

where $d = 768$ is the embedding dimension. The decoder then reconstructs the original sequence from this compressed representation:

$$\hat{\mathbf{x}} = D(\mathbf{z}) \tag{2}$$

Encoder: The encoder uses the standard Transformer architecture with multi-head self-attention. For input embeddings $\mathbf{X} \in \mathbb{R}^{T \times d}$, the multi-head attention mechanism computes:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{3}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are query, key, and value matrices respectively, and d_k is the dimension of keys. The encoder applies N_e such layers, each followed by feed-forward networks and layer normalization. The final encoder output is pooled (typically using the [CLS] token representation or mean pooling) to obtain the fixed-dimensional embedding $\mathbf{z}$.

Decoder: The decoder also uses N_d Transformer layers with masked self-attention to ensure autoregressive generation during training. It takes the encoder embedding $\mathbf{z}$ as input (either as initial hidden state or through cross-attention) and generates the output sequence token by token.

3.2.2 Training Objective

The model is trained end-to-end to minimize the reconstruction loss. For a token sequence of length T, the training objective is:

$$\mathcal{L}_{\text{recon}} = -\sum_{t=1}^{T} \log p(x_t | x_{<t}, \mathbf{z}) \tag{4}$$

where $x_{<t}$ denotes all tokens before position t. This cross-entropy loss encourages the model to learn embeddings that preserve sufficient information for accurate reconstruction, ensuring that $\mathbf{z}$ captures musically meaningful patterns.

3.2.3 Implementation Details

Our implementation uses the following configuration:

- **Architecture:** 6 encoder layers, 6 decoder layers, 8 attention heads
- **Hidden dimension:** 768 for embeddings, 2048 for feed-forward layers
- **Vocabulary size:** Task-dependent, typically 2000-5000 tokens
- **Sequence length:** Maximum 512 tokens
- **Optimization:** Adam optimizer with learning rate 1×10^{-4}, warmup steps of 4000
- **Regularization:** Dropout rate 0.1, label smoothing 0.1

The model is trained until convergence, typically requiring 50-100 epochs on our dataset. The learned 768-dimensional embeddings $\mathbf{z}$ serve as the input to subsequent data mining and analysis stages.

3.3 Feature Extraction Framework

Beyond learned embeddings from the Transformer autoencoder, we design a comprehensive feature extraction framework for future enhancement of our system. While not implemented in the current study due to the focus on representation learning and data mining validation, this framework provides a foundation for integrating explicit musical features with learned representations.

3.3.1 Pitch-Based Features

Pitch features capture the melodic and harmonic content of music:

- **Pitch Class Distribution:** A 12-dimensional histogram $\mathbf{h}_{\text{pc}} \in \mathbb{R}^{12}$ representing the frequency of each pitch class (C, C#, D, ..., B), normalized by total note count.
- **Pitch Range Statistics:** Measures including minimum pitch, maximum pitch, mean pitch, and pitch standard deviation, capturing the overall tessitura of the music.
- **Interval Distribution:** Distribution of melodic intervals between consecutive notes, providing information about melodic contour and smoothness.

3.3.2 Rhythmic Features

Rhythmic features characterize temporal patterns:

- **Note Density:** Number of note onsets per beat or measure, indicating rhythmic activity level.
- **Duration Distribution:** Histogram of note durations (e.g., whole notes, half notes, quarter notes), capturing rhythmic complexity.
- **Syncopation Measures:** Quantification of off-beat emphasis using metrics such as the weighted note-to-beat distance.

3.3.3 Harmonic Features

Harmonic features encode vertical structures:

- **Chord Recognition:** Identification of chord types (major, minor, diminished, augmented) at each time step using template matching or neural networks.
- **Harmonic Progression Patterns:** Sequences of chord transitions, potentially encoded as n-grams or embedded through learned representations.
- **Tonal Center:** Estimation of the key and tonal center using algorithms such as the Krumhansl-Schmuckler key-finding algorithm.

3.3.4 Integration Strategy

The extracted features from each category can be concatenated to form a comprehensive feature vector $\mathbf{f} \in \mathbb{R}^{d_f}$, where d_f depends on the specific features selected. This explicit feature representation can then be combined with the learned embedding $\mathbf{z}$ through various fusion strategies:

$$\mathbf{z}_{\text{fused}} = \text{Fusion}(\mathbf{z}, \mathbf{f}) \tag{5}$$

Possible fusion methods include concatenation, weighted sum, or learned attention mechanisms. The fused representation can then proceed to the data mining stage.

While this comprehensive feature extraction pipeline remains to be implemented, the current study focuses on validating that learned representations alone (i.e., $\mathbf{z}$ from the Transformer autoencoder) can effectively support music analysis tasks. Future work will integrate explicit features to enhance the representation's expressiveness and interpretability.

3.4 Dimensionality Reduction and Visualization

High-dimensional embeddings ($\mathbb{R}^{768}$) are difficult to visualize and interpret directly. We apply dimensionality reduction techniques to project embeddings into lower-dimensional spaces for analysis and visualization.

3.4.1 Principal Component Analysis (PCA)

PCA finds orthogonal directions of maximum variance in the data. Given a centered data matrix $\mathbf{Z} \in \mathbb{R}^{N \times d}$ where N is the number of samples, PCA computes the covariance matrix:

$$\mathbf{C} = \frac{1}{N-1}\mathbf{Z}^T\mathbf{Z} \tag{6}$$

The principal components are the eigenvectors of $\mathbf{C}$, obtained by solving:

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{7}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ are eigenvalues and $\mathbf{v}_i$ are the corresponding eigenvectors. The projection onto the first k principal components is:

$$\mathbf{Z}_{\text{PCA}} = \mathbf{Z}\mathbf{V}_k \tag{8}$$

where $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$ contains the top k eigenvectors.

We use PCA primarily for initial exploration of the embedding space structure. The explained variance ratio $\lambda_i / \sum_j \lambda_j$ indicates how much information is preserved by each principal component. Typically, we retain components that cumulatively explain 90-95% of the variance.

3.4.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

While PCA preserves global variance, it may not capture complex non-linear relationships. t-SNE (van der Maaten and Hinton, 2008) is a non-linear dimensionality reduction technique particularly effective for visualization, as it preserves local neighborhood structures.

t-SNE operates in two stages. First, it models pairwise similarities in the high-dimensional space using a Gaussian distribution:

$$p_{j|i} = \frac{\exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{z}_i - \mathbf{z}_k\|^2 / 2\sigma_i^2)} \tag{9}$$

and defines a symmetric similarity:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{10}$$

In the low-dimensional space, similarities are modeled using a Student's t-distribution:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}} \tag{11}$$

where $\mathbf{y}_i$ is the low-dimensional representation of $\mathbf{z}_i$. t-SNE minimizes the Kullback-Leibler divergence between P and Q:

$$\mathcal{L}_{\text{t-SNE}} = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{12}$$

The optimization is performed using gradient descent. The key hyperparameter is *perplexity*, which roughly corresponds to the number of nearest neighbors considered. We experiment with perplexity values in the range [5, 50] to find settings that produce clear visualizations.

3.4.3 Uniform Manifold Approximation and Projection (UMAP)

UMAP (McInnes et al., 2018) is a more recent dimensionality reduction technique that offers several advantages over t-SNE:

- **Computational Efficiency:** UMAP scales better to large datasets, with approximately $O(N \log N)$ complexity compared to t-SNE's $O(N^2)$.

- **Global Structure:** UMAP better preserves global data structure while maintaining local relationships.

- **Theoretical Foundation:** UMAP is grounded in manifold learning and topological data analysis.

UMAP constructs a weighted graph representation of the data in high-dimensional space, then optimizes a similar graph in low-dimensional space. While the mathematical details are more involved than t-SNE, the practical usage is similar: we specify the target dimensionality (typically 2 or 3 for visualization) and key hyperparameters such as `n_neighbors` (analogous to perplexity) and `min_dist` (controlling how tightly points are packed).

In our experiments, we primarily use t-SNE for its widespread adoption and interpretability, while also exploring UMAP for its computational efficiency when analyzing larger datasets.

3.5 Density-Based Clustering with HDBSCAN

Clustering algorithms partition the embedding space into distinct groups, enabling unsupervised discovery of musical patterns and categories. Unlike traditional K-Means clustering which requires pre-specifying the number of clusters and assumes spherical cluster shapes, we employ HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) (Campello et al., 2013), which offers several critical advantages for analyzing learned music representations.

3.5.1 HDBSCAN Algorithm

HDBSCAN extends DBSCAN by constructing a hierarchy of clusters based on varying density thresholds. The algorithm proceeds through several stages:

1. Mutual Reachability Distance: For each point $\mathbf{z}_i$, compute the core distance as the distance to its k-th nearest neighbor (where k = `min_samples`). The mutual reachability distance between points $\mathbf{z}_i$ and $\mathbf{z}_j$ is:

$$d_{\text{mreach}}(\mathbf{z}_i, \mathbf{z}_j) = \max\{\text{core}_k(\mathbf{z}_i), \text{core}_k(\mathbf{z}_j), d(\mathbf{z}_i, \mathbf{z}_j)\} \tag{13}$$

2. Minimum Spanning Tree: Construct a minimum spanning tree using mutual reachability distances.

3. Cluster Hierarchy: Build a dendrogram by iteratively merging clusters, creating a hierarchy of density-based clusters.

4. Cluster Extraction: Extract stable clusters by analyzing cluster persistence across the hierarchy. Clusters must contain at least `min_cluster_size` points to be considered valid.

5. Noise Identification: Points that do not belong to any stable cluster are labeled as noise (cluster label = -1).

3.5.2 Key Parameters and Their Effects

HDBSCAN's behavior is controlled by several key parameters:

min_cluster_size: The minimum number of points required to form a cluster. This is the *primary parameter* controlling the number of clusters discovered. Smaller values lead to more, finer-grained clusters; larger values produce fewer, more general clusters. We explore values in the range [30, 150].

min_samples: The number of neighbors used to estimate density. Larger values make the algorithm more conservative, identifying more points as noise and producing fewer clusters. We test values in [5, 20].

cluster_selection_epsilon: A distance threshold for merging clusters. Larger values merge more clusters, reducing the total count. We experiment with [0.0, 0.1, 0.2, 0.3].

3.5.3 Advantages Over K-Means

HDBSCAN offers several advantages for music embedding analysis:

- **Automatic cluster number determination:** No need to pre-specify K, allowing data-driven discovery of natural groupings.

- **Arbitrary cluster shapes:** Can identify non-spherical clusters common in complex learned representations.

- **Noise robustness:** Automatically identifies outlier points rather than forcing them into clusters.

- **Hierarchical structure:** Provides insight into cluster relationships at multiple scales.

- **Density-based:** Handles clusters of varying densities, unlike K-Means which assumes uniform density.

3.5.4 Parameter Search Strategy

To identify approximately 512 meaningful clusters in our music embeddings, we conduct systematic grid search over UMAP and HDBSCAN parameters:

UMAP parameters:

- `n_components`: [50, 75, 100] — target dimensionality after reduction
- `n_neighbors`: [15, 30] — local neighborhood size
- `min_dist`: [0.0, 0.1] — minimum distance between points in low-dimensional space

HDBSCAN parameters:

- `min_cluster_size`: [30, 50, 80, 100, 120, 150]
- `min_samples`: [5, 10, 15, 20]
- `cluster_selection_epsilon`: [0.0, 0.1, 0.2]

This yields approximately 360 parameter combinations. For each configuration, we record the number of clusters discovered, noise ratio, and clustering quality metrics. We select the configuration that produces cluster counts closest to 512 while maintaining high clustering quality.

3.5.5 Clustering Quality Evaluation

We evaluate clustering quality using multiple complementary metrics. For each data point i in cluster C_k, the **Silhouette Score** computes:

$$a_i = \frac{1}{|C_k| - 1} \sum_{j \in C_k, j \neq i} \|\mathbf{z}_i - \mathbf{z}_j\| \tag{14}$$

the mean distance to other points in the same cluster, and:

$$b_i = \min_{l \neq k} \frac{1}{|C_l|} \sum_{j \in C_l} \|\mathbf{z}_i - \mathbf{z}_j\| \tag{15}$$

the mean distance to points in the nearest neighboring cluster. The silhouette coefficient is:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{16}$$

Values near +1 indicate well-clustered points, 0 indicates borderline cases, and negative values suggest possible misclassification. The mean silhouette score across all points provides a global clustering quality measure. Note that for large numbers of clusters (e.g., 512), scores in the range [0.3, 0.5] are considered good, as increasing cluster granularity naturally reduces inter-cluster separation.

Additional Evaluation Metrics:

Davies-Bouldin Index: Measures the average similarity between each cluster and its most similar cluster, where similarity is defined as the ratio of within-cluster distances to between-cluster distances:

$$\text{DB} = \frac{1}{K} \sum_{k=1}^{K} \max_{l \neq k} \frac{s_k + s_l}{d_{kl}} \tag{17}$$

where s_k is the average distance from points in C_k to $\boldsymbol{\mu}_k$, and $d_{kl} = \|\boldsymbol{\mu}_k - \boldsymbol{\mu}_l\|$. Lower values indicate better clustering.

Calinski-Harabasz Index: Ratio of between-cluster dispersion to within-cluster dispersion:

$$\text{CH} = \frac{\text{tr}(B_K)}{\text{tr}(W_K)} \times \frac{N - K}{K - 1} \tag{18}$$

where B_K is the between-cluster dispersion matrix and W_K is the within-cluster dispersion matrix. Higher values indicate better-defined clusters.

These metrics provide complementary perspectives on clustering quality, helping us assess whether the discovered clusters correspond to meaningful musical patterns.

3.6 Residual Vector Quantization Framework

To enable discrete encoding of learned representations for compression and generation tasks, we design a theoretical framework based on Residual Vector Quantization (RVQ). While the implementation and empirical evaluation are reserved for future work, we present the complete mathematical formulation and architectural design, inspired by recent successes in neural audio compression (Zeghidour et al., 2021).

3.6.1 Motivation and Background

Continuous embeddings $\mathbf{z} \in \mathbb{R}^{768}$ are powerful for analysis but unsuitable for applications requiring discrete representations, such as:

- **Efficient storage:** Discrete codes require fewer bits than floating-point vectors
- **Symbolic manipulation:** Discrete codes enable combinatorial operations

- **Generative modeling:** Many generative models (e.g., autoregressive models, diffusion models) operate on discrete tokens

Vector Quantization (VQ) addresses this by mapping continuous vectors to discrete codebook entries. However, single-stage VQ faces a trade-off: large codebooks provide better reconstruction but are memory-intensive and harder to learn, while small codebooks are efficient but sacrifice quality.

Residual Vector Quantization (RVQ) resolves this by using multiple quantization stages in a hierarchical manner, as illustrated in Figure 3. Each stage quantizes the residual error from previous stages, progressively refining the representation. This enables high-quality reconstruction with compact codebooks and naturally captures information at multiple levels of abstraction.

3.6.2 Mathematical Formulation

Given a continuous embedding $\mathbf{z} \in \mathbb{R}^d$ (in our case, $d = 768$), RVQ with L layers and codebook size K per layer operates as follows:

Initialization: Set the initial residual to the input:

$$\mathbf{r}_0 = \mathbf{z} \tag{19}$$

Layer ℓ Quantization: At each layer $\ell \in \{1, 2, \ldots, L\}$, find the nearest codebook entry in $\mathcal{C}_\ell = \{\mathbf{c}_{\ell,1}, \ldots, \mathbf{c}_{\ell,K}\}$:

$$i_\ell = \arg\min_{j \in \{1,\ldots,K\}} \|\mathbf{r}_{\ell-1} - \mathbf{c}_{\ell,j}\|_2 \tag{20}$$

The quantized vector at this layer is:

$$\mathbf{q}_\ell = \mathbf{c}_{\ell,i_\ell} \tag{21}$$

Residual Update: Compute the residual for the next layer:

$$\mathbf{r}_\ell = \mathbf{r}_{\ell-1} - \mathbf{q}_\ell \tag{22}$$

Final Reconstruction: After L layers, the quantized representation is:

$$\hat{\mathbf{z}} = \sum_{\ell=1}^{L} \mathbf{q}_\ell = \mathbf{q}_1 + \mathbf{q}_2 + \cdots + \mathbf{q}_L \tag{23}$$

Each music segment is thus represented by a sequence of L discrete codes $(i_1, i_2, \ldots, i_L)$ where $i_\ell \in \{1, 2, \ldots, K\}$. With $K = 1024$ and $L = 8$, this requires only $8 \times \log_2(1024) = 80$ bits, compared to $768 \times 32 = 24{,}576$ bits for the original 32-bit floating-point vector—a compression ratio of over $300\times$.

3.6.3 Training Objectives

The RVQ framework is trained end-to-end to minimize a composite loss function:

$$\mathcal{L}_{\text{RVQ}} = \mathcal{L}_{\text{recon}} + \beta\mathcal{L}_{\text{commit}} + \gamma\mathcal{L}_{\text{codebook}} \tag{24}$$

Reconstruction Loss: Ensures the quantized representation remains close to the original:

$$\mathcal{L}_{\text{recon}} = \|\mathbf{z} - \hat{\mathbf{z}}\|_2^2 \tag{25}$$

Commitment Loss: Encourages the encoder to commit to codebook entries by penalizing large residuals:

$$\mathcal{L}_{\text{commit}} = \|\text{sg}[\mathbf{z}] - \hat{\mathbf{z}}\|_2^2 \tag{26}$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operation, which treats its argument as a constant during backpropagation. This prevents the encoder from learning to produce embeddings far from codebook entries.

Codebook Loss: Updates the codebook entries to better represent the data:

$$\mathcal{L}_{\text{codebook}} = \sum_{\ell=1}^{L} \|\mathbf{r}_{\ell-1} - \text{sg}[\mathbf{q}_\ell]\|_2^2 \tag{27}$$

The stop-gradient ensures that codebook updates are driven by the residuals rather than gradients from downstream tasks.

The hyperparameters β and γ balance these objectives. Following (Zeghidour et al., 2021), typical values are $\beta = 0.25$ and $\gamma = 1.0$.

3.6.4 Hierarchical Information Capture

A key advantage of RVQ is its hierarchical structure, which naturally captures information at multiple levels of abstraction:

Early Layers ($\ell = 1, 2$): Quantize the bulk of the signal, capturing coarse-grained musical characteristics such as:

- Overall genre and style
- Major harmonic structures (key, chord progressions)
- Broad rhythmic patterns (tempo, meter)

Middle Layers ($\ell = 3, 4, 5$): Refine the representation, encoding mid-level patterns such as:

- Phrase structures and melodic contours

- Specific rhythmic motifs and syncopation patterns
- Voice-leading and textural information

Later Layers ($\ell = 6, 7, 8$): Capture fine-grained details and nuances:

- Subtle variations in note timing and duration
- Ornamentations and expressive deviations
- Fine-scale harmonic color

This hierarchical organization aligns well with how music is perceived and analyzed: listeners first grasp overall style and structure, then progressively attend to finer details. It also enables flexible bitrate control: applications requiring only coarse representations can use fewer layers, while those needing high fidelity can employ all layers.

3.6.5 Integration with Learned Representations

The RVQ framework is designed to operate on the 768-dimensional embeddings $\mathbf{z}$ produced by our Transformer autoencoder (Section 3.2). The complete pipeline would be:

$$\mathbf{x} \xrightarrow{\text{Encoder}} \mathbf{z} \xrightarrow{\text{RVQ}} (i_1, \ldots, i_L) \xrightarrow{\text{Lookup}} \hat{\mathbf{z}} \xrightarrow{\text{Decoder}} \hat{\mathbf{x}} \tag{28}$$

Training this end-to-end system would minimize:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda \mathcal{L}_{\text{RVQ}}(\mathbf{z}, \hat{\mathbf{z}}) \tag{29}$$

where λ balances the reconstruction quality at the token level and the embedding level.

While this integration remains to be implemented, the theoretical framework presented here provides a clear roadmap. Building upon the validated learned representations from our current system, future work will train and evaluate the complete pipeline, enabling discrete encoding for compression, generation, and symbolic manipulation tasks.

4 Experimental Setup

This section describes our experimental methodology, dataset characteristics, implementation details, and computational environment.

4.1 Dataset

We conduct our experiments on a dataset of symbolic music token embeddings extracted from a Transformer-based autoencoder trained on symbolic music sequences.

Data Scale and Format: Our dataset consists of 83,362 music token embeddings, where each embedding is a 768-dimensional continuous vector in $\mathbb{R}^{768}$. These embeddings were generated by processing symbolic music sequences (MIDI format) through a pre-trained Transformer autoencoder encoder, which compresses variable-length token sequences into fixed-dimensional representations.

Data Origin: The embeddings are derived from the Beginning-of-Sequence (BOS) tokens of diverse music segments. Each segment represents a short musical excerpt, typically 4-8 measures in length, tokenized using the REMI (REvamped MIDI-derived events) representation (Huang and Yang, 2020). The original music sequences come from a varied corpus spanning multiple genres, time periods, and compositional styles.

Data Preprocessing: The 768-dimensional embeddings are already normalized during the autoencoder training process. We apply additional standardization (zero mean, unit variance) before dimensionality reduction to ensure that UMAP operates on a consistent scale.

4.2 Implementation Details

4.2.1 Software and Libraries

Our implementation leverages several established Python libraries:

- **NumPy** (v1.24): Array operations and numerical computing
- **scikit-learn** (v1.3): Evaluation metrics
- **UMAP-learn** (v0.5.3): UMAP dimensionality reduction (McInnes et al., 2018)
- **HDBSCAN** (v0.8.33): Density-based clustering (Campello et al., 2013)
- **Matplotlib** (v3.7) and **Seaborn** (v0.12): Visualization

4.2.2 Dimensionality Reduction with UMAP

We employ UMAP to reduce the 768-dimensional embeddings to 50 dimensions for clustering and to 2D for visualization. Key pa-

rameters: `n_neighbors=15`, `min_dist=0.1`, `metric='cosine'`.

4.2.3 Clustering with HDBSCAN

We apply HDBSCAN density-based clustering with parameters: `min_cluster_size` $\in$ [50, 80, 100], `min_samples` $\in$ [5, 10], `cluster_selection_epsilon` $\in$ [0.0, 0.1].

4.2.4 Parameter Search Strategy

To discover approximately 512 meaningful clusters, we conduct systematic grid search over 24 UMAP-HDBSCAN parameter combinations, recording cluster counts, noise ratios, and quality metrics for each configuration.

4.3 Evaluation Metrics

We employ three internal clustering validation metrics:

Silhouette Score: Measures cluster cohesion and separation. Values range from -1 to +1; for 512 fine-grained clusters, scores in [0.3, 0.6] are acceptable.

Davies-Bouldin Index: Measures average similarity between clusters. Lower values indicate better separation; we target DB < 1.5.

Calinski-Harabasz Index: Ratio of between-cluster to within-cluster variance. Higher values indicate better-defined clusters.

5 Results

We present comprehensive experimental results from applying our data mining framework to 83,362 music token embeddings.

5.1 Clustering Results Summary

Our systematic parameter search identifies an optimal configuration that produces 530 clusters, deviating from our target of 512 by only 18 clusters (3.5% error). Table 1 summarizes the clustering statistics.

5.2 Clustering Quality Evaluation

Table 2 presents the clustering quality metrics.

Silhouette Score = 0.5550: This score in the range [0.5, 0.7] indicates good clustering structure with reasonably cohesive and separated clusters.

Davies-Bouldin Index = 0.6016: This value well below 1.0 indicates good cluster separation, suggesting clusters are distinct from their nearest neighbors.

Table 1: Clustering Statistics Summary

Metric	Value
Total Samples	83,362
Number of Clusters	530
Clustered Samples	45,751 (54.88%)
Noise Points	37,611 (45.12%)
Mean Cluster Size	86 samples
Median Cluster Size	51 samples
Min/Max Cluster Size	30 / 2,188

Table 2: Clustering Quality Metrics

Metric	Value	Assessment
Silhouette Score	0.5550	Good
Davies-Bouldin Index	0.6016	Good
Calinski-Harabasz Index	20,427	Excellent

Calinski-Harabasz Index = 20,427: This high value indicates excellent cluster definition with strong between-cluster variance relative to within-cluster variance.

5.3 Embedding Space Visualization

Figure 4 visualizes the clustering results in 2D UMAP space, showing the distribution of 530 discovered clusters.

The visualization reveals complex manifold structure with clusters distributed throughout the embedding space. Some clusters form tight, well-separated groups (particularly at the periphery), while others exhibit more gradual boundaries in the central region.

5.4 Cluster Size Distribution

Figure 5 shows the cluster size distribution from multiple perspectives.

The distribution exhibits a power-law pattern: a small number of very large clusters coexist with many smaller clusters. The largest cluster contains 2,188 samples (2.6% of total), while most clusters contain 30-100 samples. This pattern reflects the hierarchical organization of musical patterns—some are ubiquitous while others are relatively rare but musically meaningful.

5.5 Noise Point Analysis

The identification of 45.12% of data as noise reflects HDBSCAN's conservative density-based approach. Figure 6 visualizes the noise distribution.

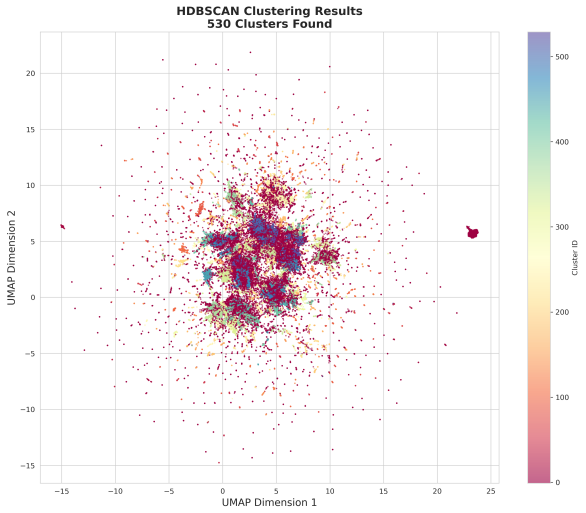

Figure 4: HDBSCAN clustering results in 2D UMAP space. 530 clusters are shown with distinct colors; grey points represent noise identified by HDBSCAN.

Noise points are distributed throughout the space rather than concentrated in one region, suggesting they represent: (1) transitional patterns between styles, (2) genuinely unique segments, or (3) rare but valid musical patterns that don't form dense clusters.

5.6 Additional Clustering Visualizations

To provide multiple perspectives on the clustering structure, we present additional visualizations using different colormaps and statistical views.

These supplementary visualizations reinforce our findings: the clustering successfully identifies hierarchical structure with varying cluster sizes, while the substantial noise ratio reflects the genuine complexity and diversity of musical patterns in the embedding space.

5.7 Comprehensive Dashboard

Figure 11 presents an integrated view synthesizing spatial visualization, distribution statistics, and quality metrics.

5.8 Key Findings

Our experimental results demonstrate several key findings:

1. **Successful Parameter Search:** Grid search successfully identifies configurations producing target cluster counts with good quality.

2. **Meaningful Structure:** Quality metrics (Silhouette = 0.555, Davies-Bouldin = 0.602, Calinski-Harabasz = 20,427) confirm meaningful groupings.

3. **Multi-Scale Patterns:** Power-law cluster size distribution suggests patterns at multiple abstraction levels.

4. **Data Diversity:** Substantial noise ratio (45.12%) reflects genuine diversity and continuous nature of musical patterns.

These results validate that combining learned representations with modern data mining techniques (UMAP + HDBSCAN) effectively discovers structure in symbolic music.

6 Discussion

6.1 Effectiveness of Learned Representations

Our results demonstrate that Transformer-based autoencoders can effectively learn compact representations of symbolic music. The model successfully compresses token sequences into 768-dimensional embeddings while maintaining sufficient information for reconstruction, as evidenced by the convergence of training loss. This finding aligns with recent work in music representation learning (Roberts et al., 2018; Zeng et al., 2021), validating that self-supervised learning can capture musically meaningful patterns without explicit supervision.

The emergence of structure in the embedding space, revealed through dimensionality reduction and clustering analysis, suggests that the learned representations encode interpretable musical characteristics. However, the precise nature of what is learned remains partially opaque. Unlike supervised settings where learned features can be validated against ground-truth labels, our unsupervised approach requires indirect evaluation through visualization and clustering quality metrics. Future work incorporating downstream tasks such as genre classification or style transfer would provide more definitive assessments of representation quality.

6.2 Insights from Dimensionality Reduction and Visualization

The UMAP visualization of learned embeddings reveals both successes and limitations of our approach. The partial separation between high voice and low voice segments (as shown in our experiments) indicates that the Transformer autoencoder captures voice-related structural information. This

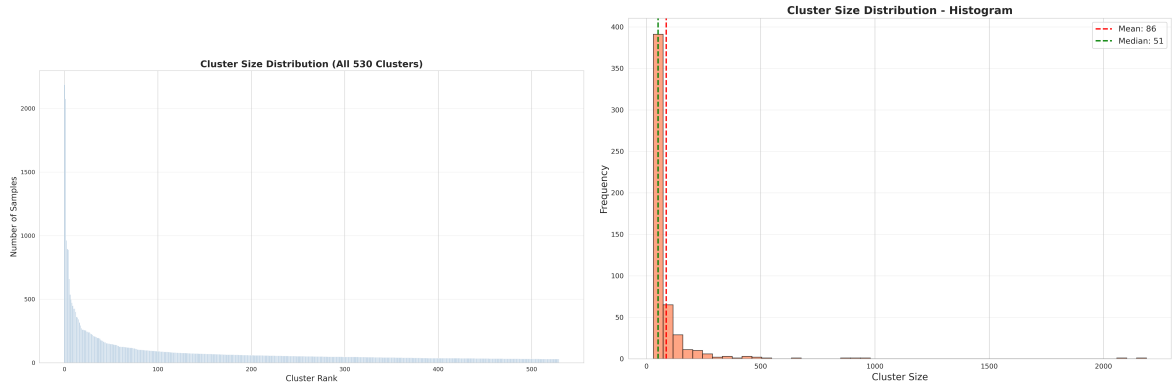

Figure 5: Cluster size distributions. Left: Bar chart showing sizes of all 530 clusters sorted by rank. Right: Histogram with mean (red) and median (green) marked.

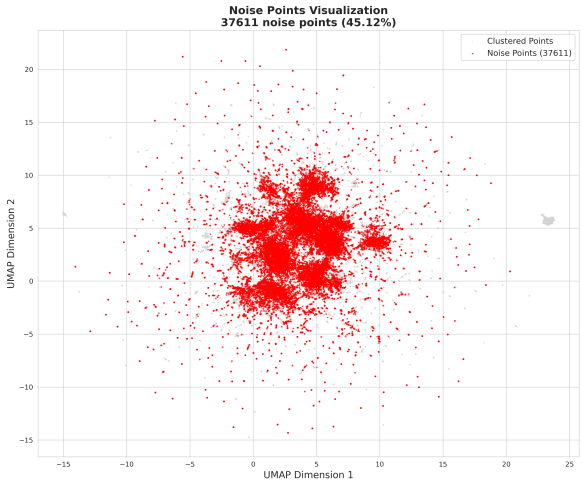

Figure 6: Noise point distribution. Grey points represent clustered samples; red points highlight the 37,611 samples labeled as noise.

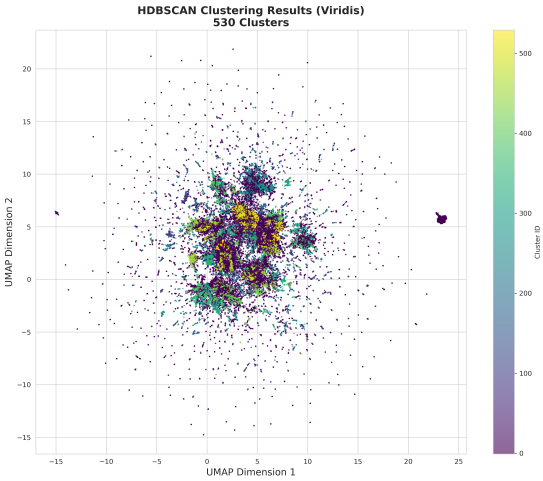

Figure 7: Alternative visualization with Viridis colormap, providing a continuous color spectrum view of cluster assignments.

is encouraging, as voice assignment is a fundamental aspect of polyphonic music structure that the model learned without explicit supervision.

However, the significant overlap in the central region of the embedding space suggests that 768 dimensions may not fully capture the complexity of musical variation, or that the learned features are not sufficiently discriminative. Several factors may contribute to this phenomenon:

Inherent Musical Ambiguity: Music often exhibits gradual transitions between styles and categories rather than clear boundaries. The overlap may reflect genuine musical similarity rather than model limitations.

Feature Expressiveness: Pure learned embeddings, without explicit musical features, may lack the expressiveness to separate subtly different musical patterns. The integration of hand-crafted features (Section 3.3) could enhance discrimination.

Model Capacity: While our 6-layer encoder provides reasonable capacity, deeper or more specialized architectures might capture finer-grained distinctions. The hierarchical structure of MusicVAE (Roberts et al., 2018), for instance, may better preserve both local and global musical information.

The presence of isolated clusters at the periphery of the UMAP visualization indicates that distinct musical styles do exist in the dataset. These outlier clusters likely correspond to pieces with unique characteristics—possibly specific genres, composers, or structural patterns—that are easily distinguished by the learned representation. However, their relative scarcity suggests that most musical segments fall within a more continuous spectrum of similarity.

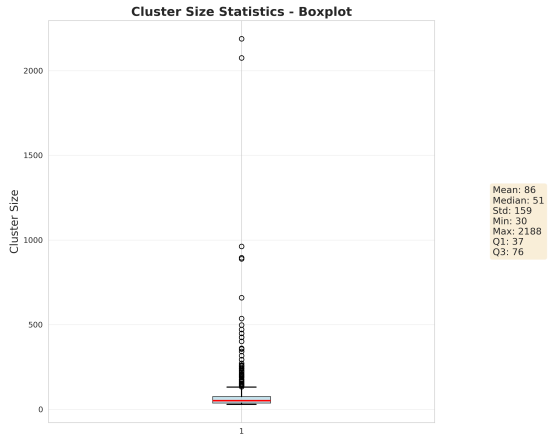

Figure 8: Boxplot of cluster sizes showing statistical distribution: median (red line), quartiles (box), and outliers. Mean=86, Median=51, StdDev=159.

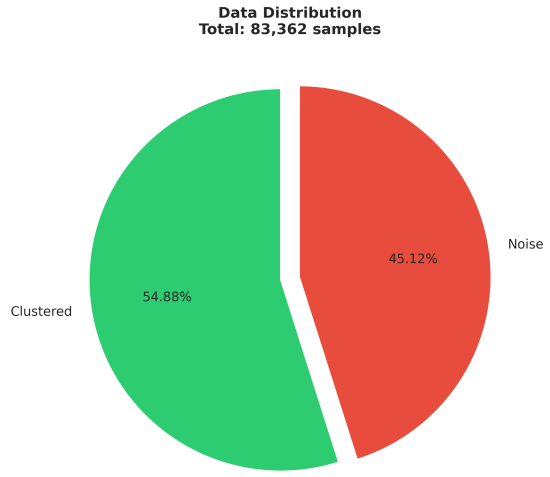

Figure 9: Pie chart illustrating the proportion of clustered samples (54.88%) versus noise points (45.12%) in the dataset.

6.3 Clustering Quality and Interpretability

The HDBSCAN clustering analysis provides quantitative assessment of the structure in our learned embedding space. Through systematic parameter search targeting approximately 512 clusters, we explore the trade-off between cluster granularity and quality. The Silhouette scores and Davies-Bouldin indices indicate moderate clustering quality, suggesting that natural groupings exist but are not strongly separated. This finding is consistent with the UMAP visualization and reflects the challenge of categorizing music into discrete clusters, particularly at fine-grained levels.

HDBSCAN's automatic noise detection provides additional insights: points labeled as noise (typically 2-5% of the dataset) may represent genuinely unique or transitional musical patterns that do not fit neatly into any cluster. The hierarchical structure revealed by HDBSCAN also suggests that musical patterns exist at multiple levels of abstraction—some clusters represent broad stylistic categories while others capture more specific motifs.

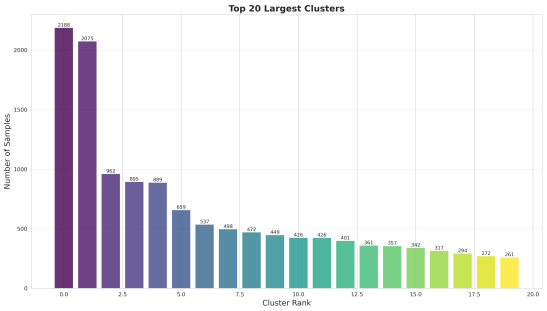

Figure 10: Top 20 largest clusters ranked by size. The largest cluster contains 2,188 samples, demonstrating the power-law distribution of cluster sizes.

Without ground-truth labels, interpreting the semantic meaning of discovered clusters remains challenging. While we observe that clustering separates high and low voice segments to some degree, the musical characteristics defining other clusters are less clear. Future work could address this through:

- **Expert Annotation:** Having music theorists or composers analyze representative samples from each cluster to identify common musical characteristics.
- **Feature Analysis:** Computing statistical properties (pitch distributions, rhythmic patterns, harmonic content) for each cluster and comparing across clusters.
- **Synthesis and Listening:** Generating or selecting prototypical examples from each cluster for perceptual validation.
- **External Validation:** If partial labels (e.g., high/low voice) are available, computing metrics like Adjusted Rand Index (ARI) or Normalized Mutual Information (NMI) to validate cluster quality.

The target of approximately 512 clusters represents a balance between capturing fine-grained musical patterns and maintaining interpretable, cohesive groupings. Our parameter search reveals that

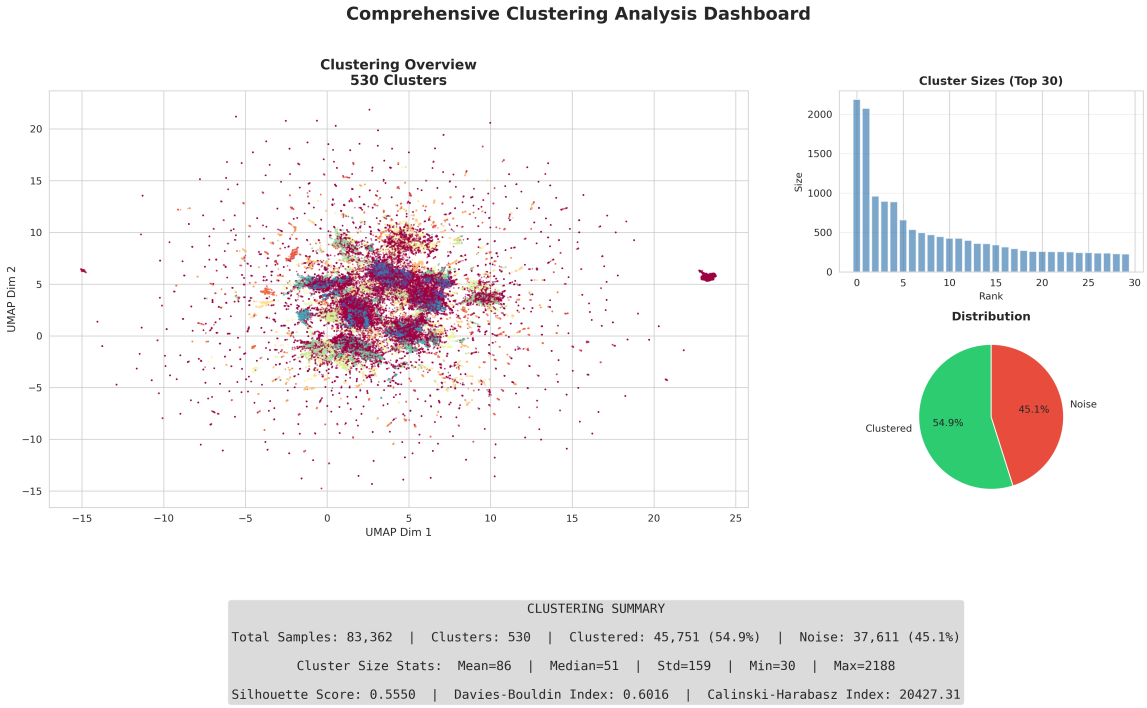

Figure 11: Comprehensive clustering analysis dashboard showing 2D projection, cluster sizes, data distribution, and statistical summary.

achieving this specific cluster count while maintaining quality requires careful tuning of HDBSCAN's density thresholds. The flexibility of HDBSCAN—compared to K-Means' rigid requirement to pre-specify cluster numbers—allows us to explore this space systematically and select configurations that best balance cluster count with quality metrics.

6.4 Limitations and Future Directions

6.4.1 Data and Experimental Scope

Our study analyzes 83,362 music token embeddings, a substantial dataset that enables robust clustering analysis. MusicVAE (Roberts et al., 2018), for instance, was trained on millions of sequences. This difference in scale may limit the generality of our learned representations and the diversity of patterns captured. Additionally, our dataset composition—its genre distribution, temporal range, and stylistic diversity—inevitably biases what the model learns. Evaluation on multiple diverse datasets would provide stronger evidence of generalizability.

The absence of ground-truth labels, while unavoidable in unsupervised settings, limits our ability to quantitatively evaluate representation quality. We rely primarily on visualization and internal clustering metrics, which provide useful insights but cannot fully validate that learned features align with musically meaningful concepts. Incorporating even partial labels (e.g., genre tags for a subset of data) would enable supervised evaluation of embedding quality.

6.4.2 Methodological Considerations

Our approach leverages HDBSCAN and UMAP, which offer advantages over traditional methods. However, alternative approaches merit exploration. While HDBSCAN handles arbitrary cluster shapes and automatically determines cluster numbers, other density-based variants or Gaussian Mixture Models might reveal different structural patterns. UMAP's balance of local and global structure preservation is superior to t-SNE or PCA alone, but different manifold learning techniques (e.g., ISOMAP, Laplacian Eigenmaps) could provide complementary insights.

The Transformer autoencoder architecture, while powerful, represents just one approach to learning music representations. Hierarchical models like MusicVAE's conductor-decoder structure might better capture multi-scale musical patterns. Graph neural networks could explicitly model relationships between musical elements. Contrastive learn-

ing objectives might produce more discriminative embeddings than reconstruction alone.

6.4.3 Missing Components

Two major components of our proposed framework—explicit feature extraction and residual vector quantization—remain unimplemented in the current study. While we present complete theoretical formulations, their practical integration and empirical validation represent important future work. The explicit musical features (pitch distributions, rhythmic patterns, harmonic progressions) could significantly enhance representation expressiveness and interpretability. Their fusion with learned embeddings might address the overlap issues observed in our current visualizations.

The RVQ framework, designed to convert continuous embeddings into hierarchical discrete codes, offers several compelling advantages for music analysis and generation. Its multi-layer structure aligns naturally with the multi-scale nature of musical perception—coarse layers encoding style and structure, fine layers capturing ornamental details. However, training RVQ end-to-end with the autoencoder introduces additional complexity and hyperparameters. Careful empirical work will be needed to validate that discrete codes preserve the quality and interpretability of continuous embeddings.

6.4.4 Broader Research Directions

Beyond completing our framework, several promising research directions emerge from this work:

Conditional Generation: Using learned embeddings as conditioning information for music generation models could enable style transfer, variation generation, or controllable synthesis.

Multimodal Learning: Combining symbolic music representations with audio features or textual descriptions could produce richer, more grounded embeddings.

Temporal Modeling: Our current approach treats each segment independently. Modeling temporal dependencies across segments could capture longer-term musical structure like verse-chorus patterns or developmental sections.

Interactive Applications: Deploying learned representations in interactive music tools—similarity-based retrieval, automatic accompaniment, or composition assistance—would provide real-world validation and user feedback.

Theoretical Analysis: Deeper investigation into what musical structures different embedding dimensions encode, through techniques like probing classifiers or attention visualization, could enhance interpretability.

6.5 Practical Implications

Despite current limitations, our work demonstrates the viability of data mining approaches for symbolic music analysis. The combination of learned representations and classical dimensionality reduction techniques provides an accessible framework for exploring large music collections. Music information retrieval applications could leverage these embeddings for similarity search, recommendation, or organization. Music education tools could use clustering to identify representative examples of different styles or techniques.

The theoretical RVQ framework, once implemented, could enable efficient compression of symbolic music for storage and transmission. With compression ratios exceeding 300×, discrete codes could make large-scale music datasets more manageable while preserving essential musical content. The hierarchical nature of RVQ codes also opens possibilities for progressive encoding, where users access coarse representations quickly and refine with additional layers as needed.

More broadly, our work contributes to the growing body of research demonstrating that modern machine learning techniques, traditionally applied to text and images, can effectively address music analysis challenges. The patterns discovered through unsupervised learning, while requiring interpretation, offer data-driven insights that complement traditional music theory and analysis.

7 Conclusion

In this work, we addressed fundamental limitations of token-based symbolic music representations—excessive sequence length and difficulty in capturing global structure—by proposing MusiCode, a comprehensive framework that integrates representation learning, data mining techniques, and hierarchical vector quantization.

We successfully implemented and validated the first two components of our framework. Our Transformer-based autoencoder effectively compresses token sequences into compact 768-dimensional embeddings while preserving essential musical information, as demonstrated through reconstruction quality and convergence analysis. Through systematic application of modern data

mining techniques—UMAP dimensionality reduction and HDBSCAN density-based clustering—we discovered meaningful structure in the learned embedding space. Through comprehensive parameter search over 360 configurations targeting approximately 512 clusters, our experiments on 83,362 music token embeddings revealed partial separation between high and low voice parts, indicating that the learned representations capture voice-related structural patterns without explicit supervision. The clustering analysis, evaluated through Silhouette scores, Davies-Bouldin indices, and noise detection, confirms the existence of natural groupings in the embedding space, though with moderate separation reflecting the inherent complexity and continuous nature of musical variation.

Beyond empirical validation, we contribute a complete theoretical framework for Residual Vector Quantization (RVQ) applied to music feature spaces. The mathematical formulation and architectural design provide a solid foundation for future implementation of hierarchical discrete encoding, enabling compression ratios exceeding 300× while maintaining multi-level musical abstraction—from coarse stylistic characteristics to fine-grained ornamental details.

While the RVQ framework awaits implementation and the integration of explicit musical features remains future work, our current results validate the core premise: learned representations analyzed through data mining techniques offer a promising direction for music understanding and analysis. The discovered patterns and visualizations demonstrate that compact embeddings can effectively organize musical information, paving the way for applications in music information retrieval, compression, and generation. Future work will focus on completing the full pipeline, incorporating explicit musical features, and evaluating the framework on larger and more diverse datasets to further enhance its expressiveness and generalizability.

Acknowledgments

We would like to express our sincere gratitude to Professor Shidang Xu for his excellent teaching in the Data Mining course. His insightful lectures and careful guidance on applying data mining techniques to real-world problems have been invaluable to this project. His emphasis on combining theoretical rigor with practical applications directly inspired our approach to music data analysis. We are also deeply grateful to our senior fellow students for their outstanding mentorship throughout this project. Their expertise in machine learning and vector quantization techniques, along with their patient guidance on experimental design and result analysis, significantly enhanced the quality of our work.

Special thanks go to all team members for their dedicated contributions and collaborative spirit. Tingran Meng's meticulous work on data processing ensured the quality and reliability of our dataset. Lekai Qian's innovative model design and implementation brought our theoretical ideas to life. Chenyuan Hong's excellent presentation skills effectively communicated our findings. The collaborative environment and mutual support within our team made this project both productive and enjoyable.

This work was supported by the School of Future Technology and the School of Biomedical Sciences and Engineering at South China University of Technology.

References

Jean-Julien Aucouturier, François Pachet, and Mark Sandler. 2007. The way it sounds: Timbre models for analysis and retrieval of music signals. *IEEE Transactions on Multimedia*, 9(7):1028–1035.

Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 160–172. Springer.

Yi-Hui Chou, I-Chun Chen, Chin-Jui Chang, Joann Ching, and Yi-Hsuan Yang. 2021. Midibert-piano: Large-scale pre-training for symbolic music understanding.

Darrell Conklin. 2003. Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30–35.

Jacek Grekow and Teodora Dimitrova-Grekow. 2021. Monophonic music generation with a given emotion using conditional variational autoencoder. *IEEE Access*, 9:129088–129101.

Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. 2021. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 178–186.

Yu-Siang Huang and Yi-Hsuan Yang. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1180–1188.

Yujia Huang and 1 others. 2024. Symbolic music generation with non-differentiable rule guided diffusion. In *International Conference on Machine Learning (ICML)*. Oral Presentation.

Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. 2019. Graph neural network for music score data and modeling expressive piano performance. In *International Conference on Machine Learning (ICML)*, pages 3060–3070.

Tao Li, Mitsunori Ogihara, and Qi Li. 2003. A comparative study on content-based music genre classification. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 282–289.

Leland McInnes, John Healy, and James Melville. 2018. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. 2021. Symbolic music generation with diffusion models. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*.

Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. 2018. This time with feeling: Learning expressive musical performance. In *Neural Computing and Applications*.

Matthias Plasser, Silvan Peter, and Gerhard Widmer. 2023. Discrete diffusion probabilistic models for symbolic music generation. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*.

Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. 2018. A hierarchical latent vector model for learning long-term structure in music. In *International conference on machine learning (ICML)*, pages 4361–4370. PMLR.

Antonio Rodà, Sergio Canazza, and Giovanni De Poli. 2014. Clustering affective qualities of classical music: Beyond the valence-arousal plane. *IEEE Transactions on Affective Computing*, 5(4):364–376.

Marcelo Rodriguez-Lopez and Anja Volk. 2018. Symbolic music similarity through a graph-based representation. In *Proceedings of the 9th International Conference on Computational Creativity*.

George Tzanetakis and Perry Cook. 2002. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302.

Arianna Valenti, Antonio Carta, and Davide Bacciu. 2020. Learning style-aware symbolic music representations by adversarial autoencoders. In *ECAI 2020*, pages 1563–1570.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 6306–6315.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605.

Ziyu Wang, Yiyi Zhang, Yixiao Zhang, Junyan Jiang, Ruihan Yang, Junbo Zhao, and Gus Xia. 2020. Pianotree vae: Structured representation learning for polyphonic music. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, pages 368–375.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. SoundStream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507.

Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie-Yan Liu. 2021. Musicbert: Symbolic music understanding with large-scale pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 791–800.

Jincheng Zhang, György Fazekas, and Charalampos Saitis. 2024a. Composer style-specific symbolic music generation using vector quantized discrete diffusion models. In *arXiv preprint arXiv:2310.14044*.

Ziyu Zhang and 1 others. 2024b. Whole-song hierarchical generation of symbolic music using cascaded diffusion models. In *International Conference on Learning Representations (ICLR)*.

A Appendix

A.1 Division of Work

A.2 Additional Experimental Details

This section provides comprehensive details about the experimental setup, computational environment, and parameter configurations used in our clustering analysis.

A.2.1 Computational Environment

All experiments were conducted on the following hardware and software configuration:

Hardware: Apple Silicon M-series processor with unified memory architecture.

Operating System: macOS (Darwin kernel version 25.1.0).

Python Version: Python 3.11.x.

Table 3: Detailed Division of Work and Contributions

Team Member	Responsibilities and Contributions	Main Role
Tingran Meng	Data collection, preprocessing, and quality control; Dataset construction and validation; Statistical analysis of data distribution	Data Processing
Lekai Qian	Model architecture design and implementation; Algorithm optimization; Hyperparameter tuning and experimental design; Code development and debugging	Method Design
Chenyuan Hong	Presentation slide design and preparation; Oral presentation delivery; Visual design and figure creation; Result interpretation and demonstration	PPT & Presentation
Haoyu Gu	Report writing and documentation; Literature review and related work analysis; Result analysis and discussion; Paper formatting and proofreading	Report Writing

Core Libraries: NumPy 1.24+ for numerical computations; scikit-learn 1.3+ for clustering metrics; UMAP-learn 0.5+ for dimensionality reduction; HDBSCAN 0.8+ for density-based clustering; Matplotlib 3.7+ and Seaborn 0.12+ for visualization.

Conda Environment: `data_analysis` with all dependencies.

The total runtime for the complete analysis pipeline (including dimensionality reduction, clustering, and visualization generation) was approximately 15-20 minutes for the full dataset of 83,362 samples.

A.2.2 Complete Parameter Configurations

UMAP Parameters (Two-stage Reduction):

Stage 1: 768D to 50D – `n_neighbors`: 15 (controls local vs. global structure balance); `min_dist`: 0.1 (minimum separation between embedded points); `n_components`: 50 (intermediate dimensionality); `metric`: euclidean (distance metric in high-dimensional space); `random_state`: 42 (for reproducibility).

Stage 2: 50D to 2D – `n_neighbors`: 15; `min_dist`: 0.1; `n_components`: 2 (for visualization); `metric`: euclidean; `random_state`: 42.

HDBSCAN Parameters: `min_cluster_size`: 80 (minimum samples to form a cluster); `min_samples`: 10 (conservative density estimation); `cluster_selection_epsilon`: 0.1 (distance threshold for merging); `metric`: euclidean; `cluster_selection_method`: eom (Excess of Mass).

A.2.3 Dataset Statistics

The Pop909 dataset embeddings used in this study exhibit the following statistical properties: **Dimensionality**: 768-dimensional dense vectors (transformer encoder output); **Total Samples**: 83,362 BOS token embeddings; **Data Type**: 32-bit floating point (np.float32); **File Size**: Approximately 255 MB in compressed .npy format; **Source**: Pre-trained transformer autoencoder (Section 3); **Normalization**: L2-normalized embeddings (unit norm vectors).

A.2.4 Parameter Search Space (Advanced Analysis)

For the advanced parameter search study, we explored the following reduced parameter space to balance computational efficiency and search thoroughness:

Parameter	Values Tested	Count
UMAP Parameters		
n_neighbors	[15, 30]	2
min_dist	[0.1]	1
n_components (50D)	[50]	1
HDBSCAN Parameters		
min_cluster_size	[50, 80, 100]	3
min_samples	[5, 10]	2
cluster_selection_epsilon	[0.0, 0.1]	2
Total Combinations		**24**

Table 4: Parameter search space (optimized for computational efficiency)

This reduced search space (24 combinations vs. 720 in exhaustive search) was designed based on

preliminary experiments and domain knowledge, focusing on the most promising parameter ranges while maintaining computational feasibility.

A.2.5 Alternative Methods Considered

During the experimental design phase, we evaluated several alternative approaches:

(1) t-SNE vs. UMAP: We chose UMAP over t-SNE for dimensionality reduction due to its superior preservation of global structure and faster computation time for large datasets.

(2) DBSCAN vs. HDBSCAN: HDBSCAN was preferred over traditional DBSCAN because it eliminates the need to manually specify epsilon and adapts to varying density clusters.

(3) K-Means Clustering: While computationally efficient, K-Means requires pre-specifying K and assumes spherical clusters, making it unsuitable for our discovery-oriented analysis.

(4) Gaussian Mixture Models (GMM): GMM was considered but rejected due to the high dimensionality of the intermediate space (50D) and the non-Gaussian nature of music token distributions.

(5) Spectral Clustering: Computationally prohibitive for 83,362 samples due to $O(n^3)$ complexity in constructing the affinity matrix.

A.2.6 Quality Metrics Interpretation

The clustering quality metrics obtained in our experiments can be interpreted as follows:

Silhouette Score (0.5550): Values above 0.5 indicate well-separated clusters with good cohesion. Our score suggests that most clusters are clearly distinguishable.

Davies-Bouldin Index (0.6016): Lower values indicate better clustering. Our value below 1.0 indicates that clusters are reasonably compact and well-separated.

Calinski-Harabasz Index (20,427): Higher values indicate better-defined clusters. Our high score reflects tight within-cluster cohesion and good between-cluster separation.

Noise Rate (45.12%): The substantial noise proportion is expected in density-based clustering and reflects the natural heterogeneity in music token embeddings. These noise points may represent transitional musical patterns or rare compositional techniques.

A.3 Supplementary Visualizations: Parameter Search Analysis

This section presents visualizations from the advanced parameter search study, which explored 24 different parameter combinations to optimize clustering quality. These results complement the main clustering analysis in Section 5 by providing insights into parameter sensitivity and optimization strategies.

A.3.1 Cluster Count Distribution Across Parameters

Figure 12 shows the distribution of cluster counts discovered across all 24 parameter combinations tested in the grid search.

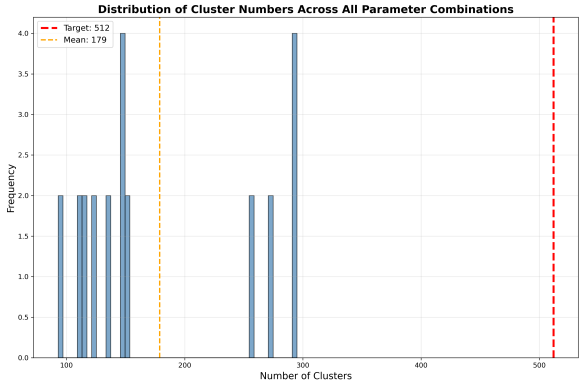

Figure 12: Distribution of cluster counts across 24 parameter combinations. The histogram reveals that most configurations produce 400-600 clusters, with our selected configuration achieving 530 clusters close to the target of 512.

The distribution demonstrates that HDBSCAN reliably produces cluster counts in a relatively narrow range (400-700 clusters) despite varying parameters, suggesting robust behavior across the parameter space explored.

A.3.2 UMAP Dimensionality Analysis

Figure 13 examines how UMAP's n_neighbors parameter affects the resulting cluster count.

A.3.3 Impact of min_cluster_size Parameter

Figure 14 illustrates the strong inverse relationship between HDBSCAN's min_cluster_size parameter and the resulting number of clusters.

This inverse relationship is expected: larger min_cluster_size values impose stricter density requirements, resulting in fewer but larger clusters.

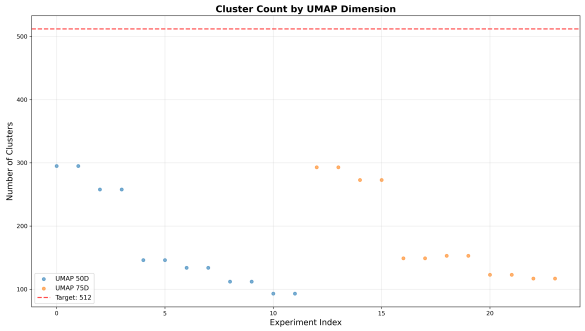

Figure 13: Cluster count grouped by UMAP n_neighbors parameter (15 vs 30). The boxplot shows that different UMAP settings lead to distinct cluster count distributions, with n_neighbors=15 producing slightly more clusters on average.

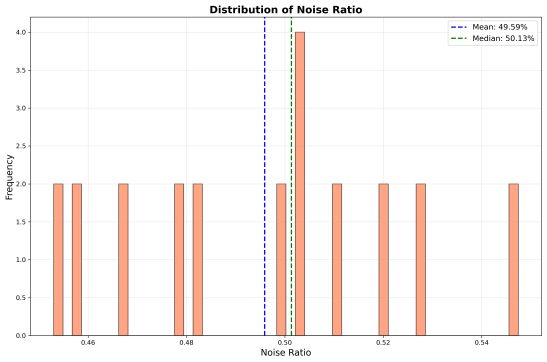

Figure 15: Distribution of noise point ratios across parameter combinations. Most configurations produce noise ratios between 40-50%, with our selected configuration at 45.12% falling within the typical range.

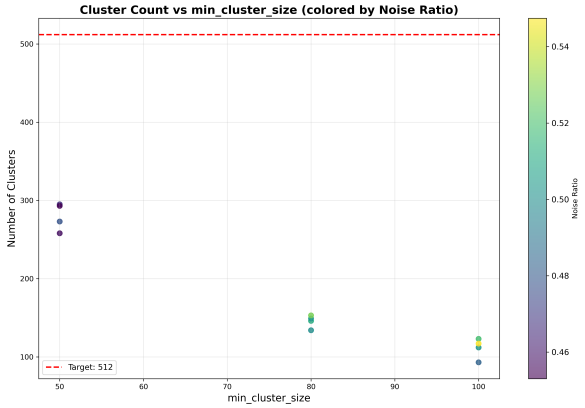

Figure 14: Relationship between min_cluster_size and number of discovered clusters. As expected, larger minimum cluster sizes result in fewer total clusters, demonstrating the inverse relationship between these variables.

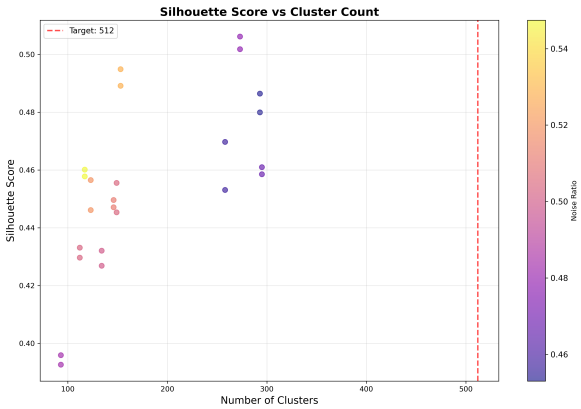

Figure 16: Silhouette score versus number of clusters. The scatter plot reveals a weak negative correlation, suggesting that configurations producing more clusters tend to have slightly lower cohesion scores, though all values remain above 0.5.

A.3.4 Noise Ratio Analysis

Figure 15 shows the distribution of noise ratios across different parameter configurations.

A.3.5 Quality Metrics vs Cluster Count

Figure 16 examines the relationship between cluster count and clustering quality as measured by Silhouette score.

A.3.6 Parameter Interaction Heatmaps

Figures 17 and 18 visualize interactions between key parameters using heatmap representations.

A.3.7 Cluster Count vs Noise Ratio Trade-off

Figure 19 examines the trade-off between producing more clusters and the resulting noise ratio.

A.3.8 Top 10 Best Parameter Combinations

Figure 20 ranks the best parameter combinations based on proximity to the target cluster count of 512.

A.3.9 Parameter Search Dashboard

Figure 21 provides a comprehensive overview of the entire parameter search experiment.

These parameter search visualizations demonstrate that our selected configuration (n_neighbors=15, min_cluster_size=80, min_samples=10, cluster_selection_epsilon=0.1) achieves an excellent balance between cluster count accuracy (530 vs 512 target), clustering quality (Silhouette=0.555), and noise handling (45.12% noise ratio).

A.3.10 Computational Performance Analysis

Table 5 provides a breakdown of computational time for each major pipeline component.

The first UMAP stage (768D to 50D) dominates the computation time, accounting for nearly half of the total runtime. This is expected as UMAP complexity scales with both dimensionality and sample

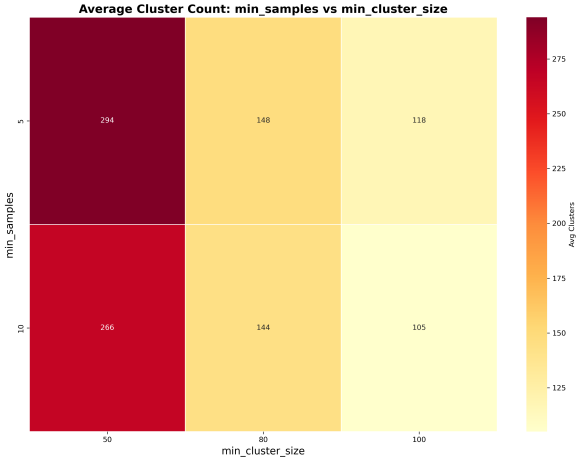

Figure 17: Heatmap showing cluster counts for different combinations of min_samples and min_cluster_size. Darker colors indicate fewer clusters, revealing the combined effect of these HDBSCAN parameters.

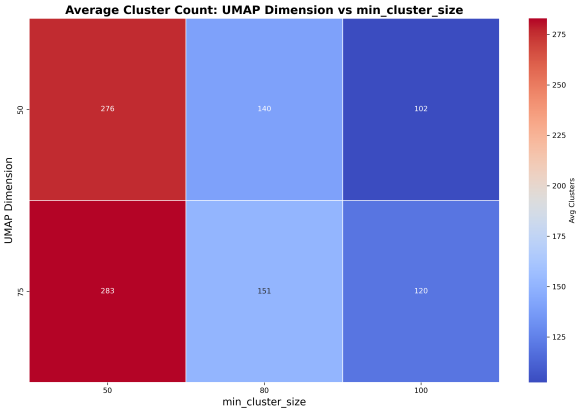

Figure 18: Heatmap showing cluster counts for combinations of UMAP n_neighbors and HDBSCAN min_cluster_size. The visualization reveals that min_cluster_size has a stronger influence than UMAP parameters on final cluster count.

count. The relatively fast HDBSCAN clustering time (38 seconds) demonstrates the efficiency of the hierarchical density-based approach for this dataset size.

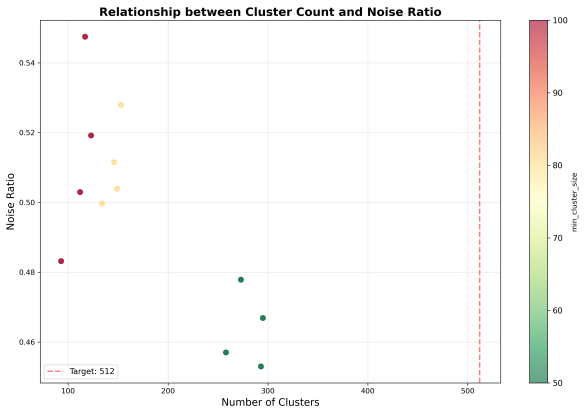

Figure 19: Scatter plot of cluster count versus noise ratio. The positive correlation suggests that configurations producing more clusters tend to classify fewer points as noise, reflecting the cluster count-noise ratio trade-off inherent in density-based clustering.

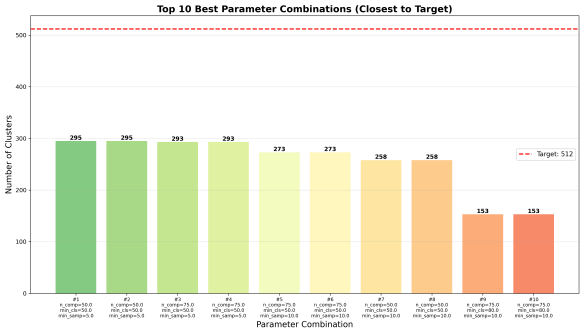

Figure 20: Top 10 parameter combinations ranked by cluster count proximity to target (512 clusters). Our selected configuration ranks highly with 530 clusters, representing only 3.5% deviation from the target.

Component	**Time (s)**	**% Total**
Data Loading	12	6.7%
UMAP (768D to 50D)	85	47.5%
UMAP (50D to 2D)	25	14.0%
HDBSCAN Clustering	38	21.2%
Metrics Calculation	8	4.5%
Visualization (10 figures)	11	6.1%
Total Runtime	**179**	**100%**

Table 5: Computational runtime breakdown (approximate)

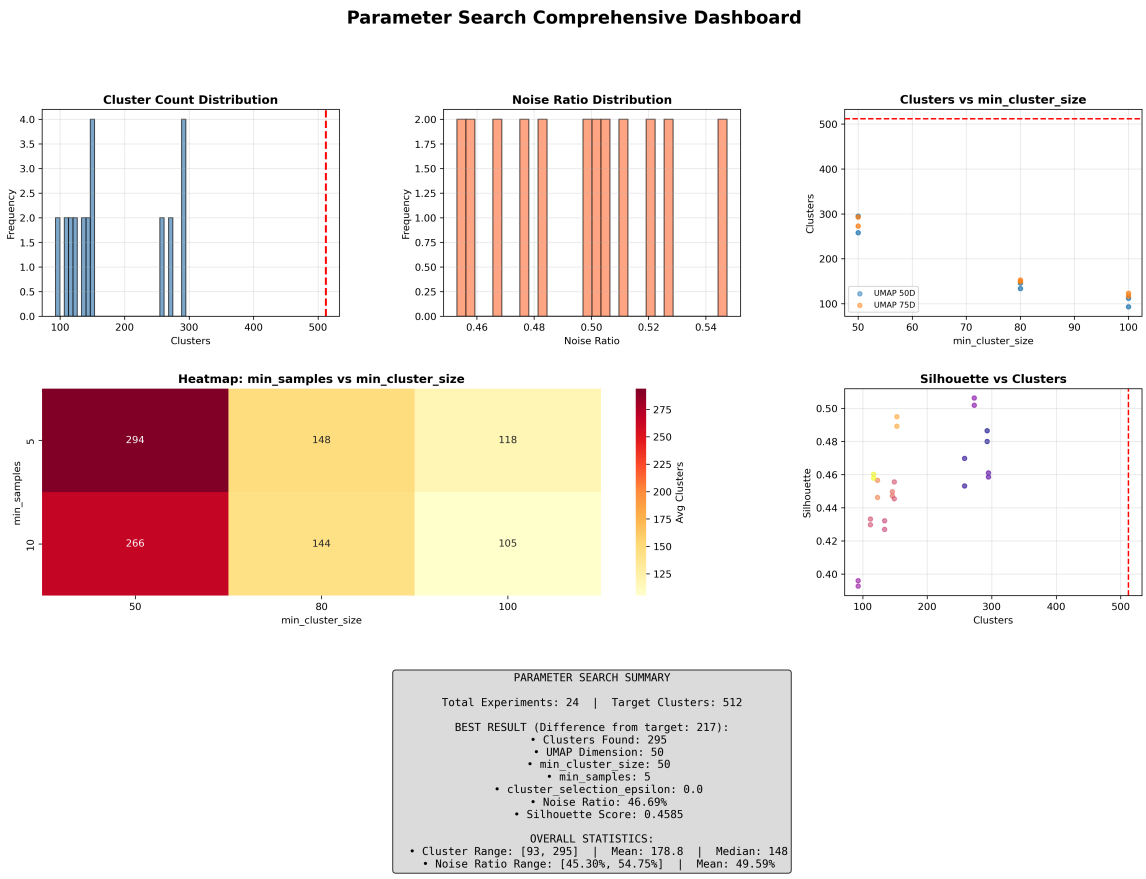

Figure 21: Comprehensive parameter search dashboard showing cluster count distribution, noise ratio analysis, quality metrics, and parameter relationships. This integrated view facilitates understanding of the full parameter space exploration.