Expert-MusiComb: Injective Domain Knowledge in a Neuro-Symbolic Approach for Music Generation

Lorenzo Tribuiani*, Luca Giuliani*, Allegra De Filippo and Andrea Borghesi

Department of Computer Science and Engineering (DISI), University of Bologna

Abstract

The significant expansion of data-driven technologies in the past decade has highlighted the crucial role of structured data, given the more relevant and meaningful informative content that they can provide to artificial intelligence (AI) applications. Nonetheless, there are domains based on inherently unstructured data, such as the audio domain. In those cases, the possibility of relying on an automated system capable of extracting structured features from raw data could serve as a pivotal element in enhancing and strengthening the capabilities of an AI system. In this work, we propose an automated feature extractor which leverages machine and deep learning methodologies to retrieve two higher-level musical attributes from short MIDI samples, namely the harmonic content of the sample – through its chords progression – and the role that such sample could have within a multi-track composition – i.e., melody, bass, or accompaniment. We perform our tests on a dataset containing ground truth information to assess quantitative results and later integrate our models within the state-of-the-art framework for combinatorial music generation *MusiComb* to check for harmonic and melodic consonance on the downstream generative task.

Keywords

Music Generation Systems, Generative AI, Chord Prediction, Machine Learning, Constraint Programming

1. Introduction

Computer-aided music generation combines computer science, machine learning, and music theory to compose, produce, or assist in creating music. This interdisciplinary field poses a significant challenge due to the complex mix of creativity, emotion, and technical requirements involved, making it one of the most demanding tasks for AI to undertake. MusiComb, originally conceived as an implementation of the work theorized by Hyun et al. in the paper [1], emerges as a framework for combinatorial music generation. It employs Constraint Programming to generate the final piece, while utilizing deep learning and machine learning techniques for data preparation and generation. Through the fusion of short MIDI samples, this system excels at crafting well-structured compositions and empowers users by allowing them to shape the creative process through the customization of various music-related parameters. As well summarized in the related paper [2]: *it represents a novel music generation approach aimed at overcoming generative model limitations, by properly combining a set of samples under user-defined constraints.*

MusiComb, alongside *ComMU*, the MIDI sample dataset introduced in [1] and primarily utilized during the initial development of the framework, has established a standardized set of significant features of harmonic and structural attributes of each sample within the dataset. These features serve as fundamental components utilized by the framework for sample combination and music generation. While ComMU serves as an exemplary dataset for the tasks, the imperative for new datasets has become evident. This necessity arises not only to incorporate fresh samples but also to furnish MusiComb with a broader array of potential features, such as *music genre*, enabling users to explore a wider spectrum of potential outcomes. This, coupled with the challenge of locating MIDI datasets labeled consistently with ComMU, underscored the necessity for an automated feature extractor. Our primary focus lies in

CREAI 2024 – Workshop on Artificial Intelligence and Creativity, Oct.19 – Oct.24, 2024, Santiago de Compostela, Spain
*Corresponding author.
lorenzo.tribuiani@studio.unibo.it (L. Tribuiani); luca.giuliani13@unibo.it (L. Giuliani); allegra.defilippo@unibo.it (A. De Filippo); andrea.borghesi3@unibo.it (A. Borghesi)
0000-0002-9120-6949 (L. Giuliani); 0000-0002-1954-7271 (A. De Filippo); 0000-0002-2298-2944 (A. Borghesi)

estimating a subset of features that are not readily accessible or discernible through the MIDI protocol. These features, such as *track roles* and *chord progressions*, pose greater complexity in estimation due to their indirect nature. They hold significant relationships with the harmonic and structural attributes of the samples. Establishing correlations between known properties or elements of the sample and these desired features necessitates the use of machine learning and deep learning systems. Moreover, evaluating the results, particularly in the chord progression domain, presents challenges as existing systems often rely on human intervention.

The concluding phase will focus primarily on the modifications and additions implemented within the MusiComb framework, aiming to address various aspects and limitations inherent in the system itself. A key addition involves an automatic sample extraction algorithm designed to extract small, repetitive sequences (samples) from complex and lengthy MIDI files, facilitating the integration of new datasets. Additionally, to enhance the flexibility of sample selection mechanisms and mitigate their inherent rigidity based on user-selected parameters, minor adjustments have been incorporated into the pipeline. The introduction of new MIDI datasets, coupled with these modifications, is intended to render the framework more adaptable and operate more flexibly.

This approach enables a broader range of possible combinations, thereby reducing the deterministic nature of the overall process relative to the initial set of user parameters. Further elaboration on the rationale for those approaches will be provided in subsequent discussions, specifically: Section 2 will offer a comprehensive overview of relevant existing works and the main rationale behind this work; Section 3 will delve deeply into the challenges and complexities associated with feature extraction and the main problematic aspects and difficulties encountered in addressing these challenges; Section 4 will introduce the primary integrations and modifications made to the MusiComb framework, offering a detailed explanation of the sample extraction algorithm and the key advantages resulting from the adjustments to the sample selection pipeline, particularly in relaxing selection criteria. Some of the compositions generated by Expert-MusiComb can be heard at the following link: https://soundcloud.com/lorenzo-tribuiani/sets/musicomb.

2. Background and Motivation

The field of computer-aided music generation has recently experienced a significant advancement in the use of end-to-end neural systems. Transformer-based models such as Music Transformer [3] paved the way to more successful projects like MusicLM [4], Jukebox [5], Noise2Music [6], and even professional tools like Suno. However, the adoption of these models introduces several drawbacks. Among all, these systems are still subject to abrupt timbre changes and noisy outputs, which restrict their professional use as well as offering very limited user control and requiring high computational demands, thus preventing them from real-time use [7]. On top of that, the inherently opaque nature of neural architectures has been proven to lead to unintended plagiarism, with a consequent lack of recognition of the human artists whose compositions have been used to train the models.

Driven by the goal of addressing these challenges, there is renewed interest in symbolic-based generative models within the research community. Traditionally, Probabilistic and Hidden Markov Models (HMMs) have been widely used for both chord and melody generation. For example, [8] uses an HMM for Bach-inspired chorale harmonies, [9] involves pattern recognition and recombination techniques to create compositions that replicate th style of various classical composers while [10] and [11] apply complex graphical models to melody harmonization. More recently, systems such as Morpheus [12] and GEDMAS [13] employed explicit rules and probabilistic methods to generate melodies or entire tracks according to certain constraints, while Pachet et al. [14] combined symbolic models with neural architectures to exploit the power of both frameworks. Our former work, MusiComb [2], aligns as well within this research area, as it employs a combinatorial approach to music generation although it focuses on the arrangement of predefined segments of notes (samples) rather than generating music on a note-by-note basis.

The main strength of sample-based music composition comes from its higher compatibility with contemporary pop music compositional and production standards since, over the past thirty years, the introduction of samplers and Digital Audio Workstations (DAWs) has significantly shifted the music industry's workflow towards extensive use of sample libraries [15]. However, given that the arranged samples must meet specific properties and constraints to harmonically integrate into longer sequences, the challenge of correctly locating them rapidly became problematic and task-intensive, especially as the size of the databases started to expand [16]. Current sample libraries often provide some metadata such as key signature and tempo, along with additional high-level labelling that can be used to create filters, but information about chord progressions, instrument type, and track role is most of the time lacking, hence preventing a fully automated procedure. Similarly, research in sample-based music generation is restricted by technical limitations, as major datasets for synthetic music generation such as the Lakh MIDI Dataset [17] or MuseDB18 [18] lack this critical information. For this reason, our aim is to build an automated pipeline that could work as well with larger datasets by employing machine learning models to extract this kind of metadata whenever it is missing.

3. MIDI Feature Estimation

The core challenge of this work revolved around MIDI feature estimation. As previously mentioned, MusiComb established a standardized set of features for each sample, essential for the proper functioning of the framework. Specifically, eight distinct features were identified: *Beats Per Minute* (BPM), *number of measures*, *key signature*, *genre*, *track role*, *chord progression*, *time signature*, and *rhythm*. These eight features have been categorized into two main groups for clarity:

Direct features including BPM, number of measures, key signature, genre, time signature, and rhythm, refer to those characteristics whose values are either explicitly written or easily extractable from the MIDI data itself. Key signature can indeed present a minor obstacle in estimation, as it may not always be explicitly encoded within the MIDI data. However, there are existing algorithms capable of estimating the key signature with a reasonable level of confidence, such as the Krumhansl-Schmuckler algorithm utilized in this study. Also, is commonly assumed that genre can be implicitly inferred from the properties of the dataset itself.

Indirect features such as *track-role* and *chord progression*, are attributes that are unlikely to be explicitly encoded within the MIDI protocol of the sample.

The primary challenge in this study is to handle indirect features. While track-role estimation can be addressed using classical classification techniques (i.e. SVM), chord progression estimation differs fundamentally. This calls for alternative methods, such as using GRU layers typically employed for Natural Language Processing tasks, and different evaluation metrics to effectively tackle these tasks.

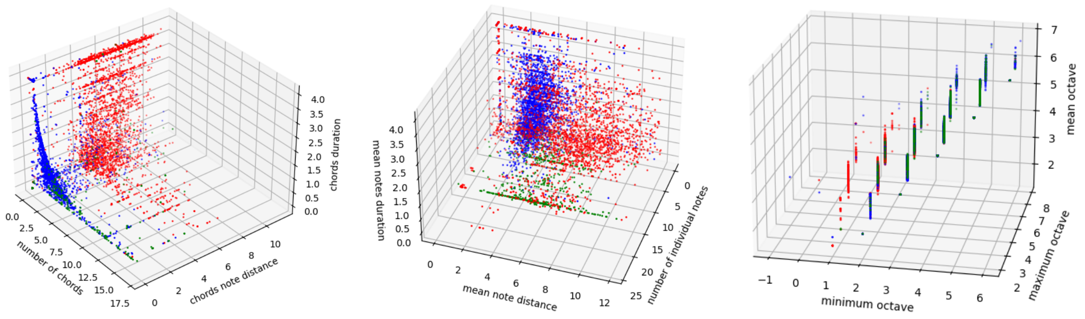
3.1. Track Role Estimation

The track role (i.e., the function of a sample within a music piece) is challenging to estimate due to its contextual nature. Defining track roles often relies on human interpretation and overall musical context, posing challenges in establishing clear class boundaries without nuanced distinctions. Another challenge arises from the similarity between track roles, with MusiComb standardizing six distinct classes: *main melody*, *sub melody*, *riff*, *accompaniment*, *pad*, and *bass*. Some roles, like main and sub-melody, share similar concepts and structures, making the classification process harder. The six-track role classes can be grouped into three primary macro-groups: *Melody*, *Accompaniment*, and *Bass*, as shown in Table 1. This grouping highlights structural similarities within each macro-group, complicating their differentiation. Additionally, the riff class exhibits similarities with both melody and accompaniment, as illustrated in Fig. (1).

In our study, significant effort was devoted to identifying a feature set suitable for training a **Support Vector Machine**, due to a preliminary empirical evaluation we made of different ML algorithms, for classifying six distinct MusiComb track roles. We focus on structural musical elements that vary across classes while maintaining consistency or similarity within each class. The feature set for classification

Melody	Main melody
	Sub melody
	Riff
Accompaniment	Accompaniment
	Pad
Bass	Bass

Table 1
Track role classes. Elements within the same macro-group (melody, accompaniment, and bass) exhibit similarities.



(a) Chord related features (b) Notes related features (c) Octave related features

🟥 Accompaniment 🟦 Melody 🟩 Riff

Figure 1: In the feature space related to chords, as shown in Figure 1a, there is a clear boundary separating Accompaniment and Melody-like tracks. However, no distinct boundary can be observed for Riffs. Conversely, Figure 1b demonstrates that note-related features exhibit a less pronounced but more discernible separation among the three classes. Lastly, Figure 1c illustrates that octave-related features provide a good distinction between Accompaniment-like and Melody-like tracks, yet they do not offer a clear differentiation between Melody and Riff.

was selected from those directly accessible from the MIDI protocol or modified versions thereof, ensuring their availability in external datasets. Since track role definition is independent of harmonic properties, only structural features were used for classification. In a data-informed approach, using personal domain knowledge, and based on the primary characteristics of each group, a set of eleven independent features was chosen for classification.

1-2. Mean chords number & Mean notes number: The number of chords and individual notes is crucial for the track role. We normalize them to mitigate the impact of sample length differences, ensuring a consistent representation of chord and note densities across samples.

$$mean\_chords\_number = \frac{chords\_number}{measures\_number}$$

$$mean\_notes\_number = \frac{notes\_number}{measures\_number}$$

3-4. Chords duration & notes duration: Longer durations indicate greater importance in the score, potentially influencing classification. To maintain consistency across samples of different lengths, durations are normalized based on the number of measures.

5. Chords note distance: Not all simultaneous note sets adhere to conventional chord definitions[1]. Hence, we consider the distances between notes within these groups, leveraging the consistent

[1]Some instances, like those with fewer than three notes, serve to reinforce a melody or enrich harmony.

ratios between note distances in chord modes well-known in music theory. We use a modulo 12 representation for note distances, disregarding octave information, to maintain consistent distance measurements across octaves.

6. **Notes distance**: In accompaniments, chords can be played individually or as arpeggios, complicating classification. To address this, we use the mean distance between individual notes. This metric, expressed modulo 12, helps differentiate chord arpeggios based on their mean distances.

7. **Number of chord's note**: A single chord provides insights into the sample's role. While traditionally three notes define a chord, fewer may indicate bicords, and three or more suggest various structural elements. This value is represented as the mean number of notes per chord.

8-9-10. **Minimum octave, maximum octave and mean octave**: Minimum and maximum octaves set pitch variability boundaries within the track. The mean octave offers a central reference, emphasizing the most relevant octave in the piece. Recognizing these boundaries aids in excluding certain track roles based on expected octave characteristics.

11. **Instrument**: Instrument features are included to account for their association with specific track roles or octave ranges.

3.1.1. Preprocessing

The ComMU dataset, the only dataset including all the metadata needed for the functioning of MusiComb, was used for training. Preprocessing involved extracting the 11 relevant features and making adjustments for robustness against minor variations. All data vectors were scaled using the *z-score* equation[2] to standardize the range, preventing larger-ranged data points from disproportionately influencing the SVM model.

3.1.2. Training and results

The SVM model was trained on the *ComMU* dataset and fine-tuned using *Grid Search* with *cross-validation* to identify the best hyperparameters. Grid search was performed on three **kernel** types: *RBF*, *polynomial*, and *linear*, with five **C** values (0.5, 1, 5, 10, 100) and two **tolerance** values (0.01 and 0.001). The dataset was split into five subsets for cross-validation. Table 2 shows the top 5 SVM models, including their parameters and performance on each dataset subdivision, together with a K-Nearest Neighbours baseline.

Model	Kernel	C	Tolerance	TAS1*	TAS2	TAS3	TAS4	TAS5	Mean test score	Mean F1 score
SVM	RBF	10	0.001	**0.81**	**0.81**	0.81	**0.81**	**0.82**	**81.1%**	**78.6%**
SVM	RBF	10	0.01	0.81	0.81	**0.82**	0.81	0.81	81.08%	78.6%
SVM	RBF	5	0.001	0.79	0.8	0.81	0.81	0.81	80.53%	77.9%
SVM	RBF	5	0.01	0.79	0.8	0.81	0.81	0.81	80.53%	77.6%
SVM	Poly	10	0.001	0.8	0.79	0.81	0.79	0.81	79.89%	76.8%
KNN									57%	50%

* Test Accuracy Score for the given dataset subdivision

Table 2
Results from dataset splits and averages for the top 5 scoring results and KNN baseline.

The SVM model outperformed the KNN baseline, achieving an overall accuracy of 81.1% and an F1 score of 78.9%. The analysis of the confusion matrix indicates that the lower F1 score is mainly due to the sub-melody and riff classes. Discriminating the riff class is challenging, especially in distinguishing

[2] $z = \frac{X-\mu}{\sigma}$, where X is the current data point, μ is the mean and σ the standard deviation.

it from the main melody and accompaniment. The sub-melody class is frequently misclassified as the main melody, while the main melody is sometimes incorrectly classified as sub-melody. This unexpected behaviour could be influenced by the class distribution in the training set. Figure 2 illustrates that the

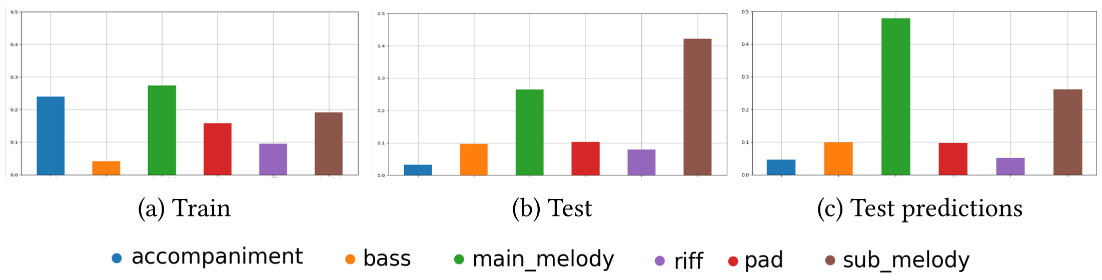


Figure 2: Distribution of classes in the training set, test set, and SVM predictions on the test set.

main melody is more prevalent in the training dataset (2a), and the model has replicated this distribution in its predictions (2c) on the test set[3].

3.2. Chord Progression Estimation

Chord progression estimation involves finding the ordered sequence of chords that accompany a melody. It relies heavily on harmonic elements such as notes and pitches, leading to variations based on the sample type.

1. Chord progressions can be non-unique, influenced by the track role of a sample. Melody-like samples present challenges as a single melody can harmonically match multiple chord progressions, each producing a unique sound. In contrast, accompaniment and bass-like samples are typically based on specific chord progressions, requiring a stricter classification approach. For chord progression estimation, the macro groups from Table 1 will be used to define the three primary classification domains.
2. The classification task has an extensive solution space defined by the unique possible chord names, resulting in thousands of potential chord combinations based on harmonic rules. Balancing the retention of harmonic information by simplifying the space is essential for ease of classification.
3. A harmonic metric tailored for chord progression estimation in melody samples is missing from the literature. While classification accuracy is important, ensuring harmonic soundness in the final results is crucial. A metric emphasizing harmonic aspects over mere accuracy is required.

Point 1 suggests training a unique model for each of the three macro groups (Melody, Accompaniment, and Bass) using a **Shared Model Split Weights** approach. Although there are significant differences between track roles, the estimation tasks are similar. Respectively, points 2 and 3 lead to a simplification of the labels based on their names described in Section 3.2.1 and the adoption of a metric emphasizing harmonic soundness, inspired by the work on *Mathematical Harmony Analysis* [19]. Moreover, understanding the temporal or positional context of the sample representation is crucial. Both samples and chord progressions can be viewed as *time series* and a time-aware representation is crucial. Building on the work of *Hyungui et al.* [20], recurrent neural networks based on GRU layers combined with an autoregressive pipeline have been selected to address this task.

3.2.1. Preprocessing

We adopted a modified version of the *Pitch Class Vector (PCV)* representation from [20] for feature preprocessing in our study. The PCV consists of 12 bins representing the 12 possible notes; we scale these values using normalized Velocity[4] to reintroduce accent information by emphasizing notes played

[3]The dataset split is the same adopted for ComMU and indicated in the dataset's metadata.
[4]Velocity is a MIDI parameter that indicates the force applied when pressing a key on a MIDI keyboard. Normalized velocity is a scaled representation in the [0,1] range.

with greater force. Fig. 3 illustrates the modified PCV for a MIDI sample.

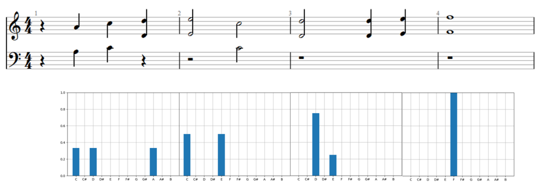


Figure 3: Modified PCV of a MIDI sample.

original	simplification
major	major
minor	minor
diminished	/[*]
augmented	major
alterations (7th, 8th, ...)	major/minor

[*] Due to their dissonant nature, diminished chords have been excluded from the dataset.

Table 3: Chords name simplification rules.

Conversely, a classical approach using one-hot encoding has been employed for representing chord names. As illustrated in table 3 all chord names present in the ComMU dataset have been mapped to just two categories: **Major** and **Minor** chords. This reduction narrows down the solution space to 24 possible elements. Finally, **data augmentation** was applied to the dataset by transposing samples and their corresponding chord progressions across various note intervals, resulting in 11 additional entries for each sample. This technique has been fundamental in developing the chord progression estimator. This is primarily because transpositions of the same sample should result in corresponding transpositions of the chord progression output, enhancing the system's robustness. Injecting this knowledge is neither direct nor easy; it is something the model must learn during training. This approach offers a dual benefit: it increases the dataset size by augmenting the existing samples, a technique known to improve *generalization*, and it incorporates *transposition invariance* into the model during training.

3.2.2. Model Architecture

The model architecture uses an *Autoregressive* approach with GRU layers to capture temporal patterns. For a sample with N measures, each measure is analyzed independently to reconstruct temporal information focusing on chords. The input structure is as follows: **Initial conditions**, representing the encoding of the three previous chords at time steps $t-1$, $t-2$, and $t-3$ where all values are set to zero for the first measure (START token), **measure n**, the Pitch Class Vector (PCV) for the considered feature and **measure n+1**, the features of the next measure to provide information on the harmonic direction. Individual learned embeddings are used for each feature, and the final model input is a tensor obtained by concatenating these embeddings, resulting in a shape of $(5, 64)$.

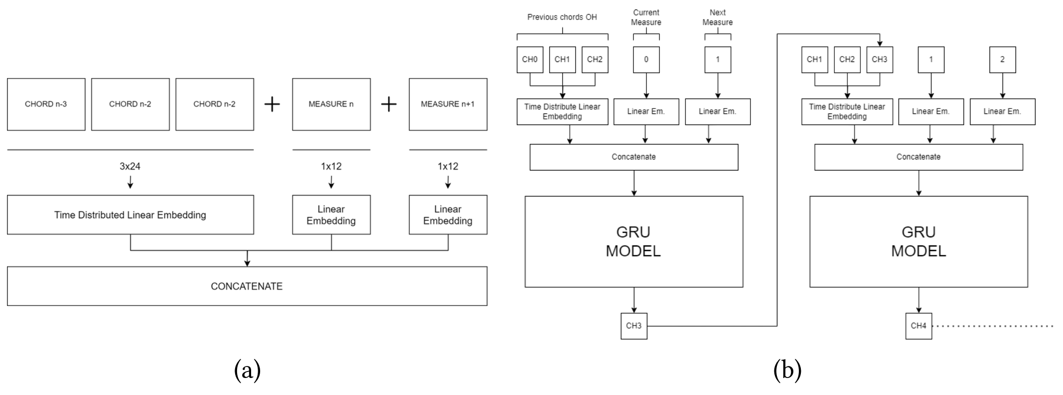


Figure 4: Visual representation of model inputs (4a) and the model's autoregressive pipeline (4b).

The main model architecture features a modular design, with each module consisting of a **BiGRU** layer with 128 cells and *hyperbolic tangent* activation, followed by **Layer Normalization** and **Dropout** with a probability of 0.3. In this specific application, two modules were utilized with a final classification head included in the model. Figure 4b illustrates the model's autoregressive pipeline. The one-hot encoding of each chord is added to the initial conditions as the window shifts forward, allowing the model to use current and past chord information.

3.2.3. Training & Results

The model predicts chords for each 4-sized window of samples. The ComMU dataset was augmented and split into three datasets based on track-role macro groups: *melody*, *accompaniment*, and *bass*. The model was trained thrice on each dataset with consistent hyperparameters[5]. Table 4 presents the results of the best model weights evaluated on the corresponding test set for the specific task.

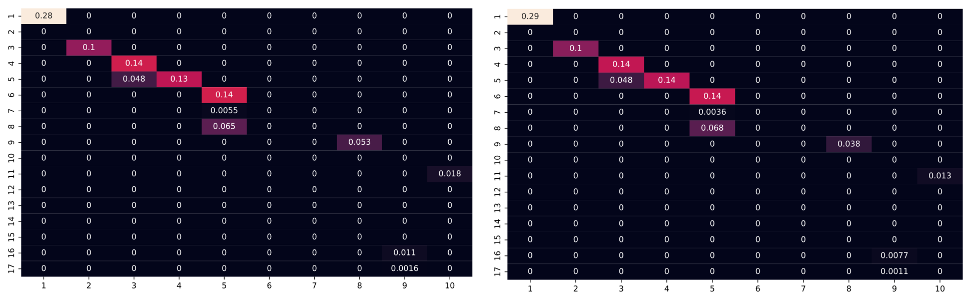
	Test Accuracy	Test F1 Score
Accompaniment Model	0.7511	0.7243
Melody Model	0.6297	0.6134
Bass Model	0.7514	0.7252

Table 4
Optimal results for the chord progression estimation task.

The accompaniment and bass models performed better than the melody model, which still achieved a solid 63% accuracy and 61% F1 score. The confusion matrix highlighted frequent misclassifications, especially between minor and specific major chord sequences, known as *relative minors.* This emphasizes the importance of a metric focusing on harmonic aspects. In [19], a general rule describing the pleasantness of a note interval is introduced: *two notes played together sound harmonious (consonant) when their frequency ratio uses small whole numbers.* Building upon this concept, we developed a more robust metric by examining the distribution of frequency ratio values across the entire dataset for both labels and predictions. Particularly, being n the number of measures in a sample, $notes(n)$ the set of frequencies of notes in that measure and $chord(n)$ the set of frequencies of notes of the chord of that measure, we define the set of *frequency ratios* for a measure as:

$$ratios(i) = \left\{ \sim \frac{a}{b} \mid \forall\, a \in notes(i),\ \forall\, b \in chord(i) \right\} \quad for \quad i \in [0, \dots, n]$$

These two distributions, represented as matrices of numerator-denominator values, were then evaluated in terms of Cosine Similarity.



(a) (b)

Figure 5: Example of frequencies-ratio distribution in matrix representation (Melody class) for both labels (5a) and predictions (5b).

Figure 5 displays the *Frequency Ratio Distribution (FRD)* matrices for both labels and predictions, while Table 5 presents the cosine similarities between these distributions for all individual tasks. As

[5]**Initial LR**: 10^{-3}, **minimum LR**: 10^{-7}, **LR scheduler**: OnPlateau, **Optimizer**: AdamW, **Epochs**: 100, **Batch size**: 1024.

	FRDs Cosine Similarity
Accompaniment Model	0.994
Melody Model	0.986
Bass Model	0.99

Table 5
FRDs Cosine similarity for each model's weights in the specific classification task.

observed, the model has adjusted its weights to closely replicate the frequency ratio distribution of the original labels. Given the assumption that the chord progressions labelled in the dataset sound good, mimicking this distribution indicates the quality of the model's predictions which is, thus, as good as the labelled one.

4. Expert-MusiComb

All the technologies discussed in Section 3 were applied to extend MusiComb, especially for *dataset expansion*. Additionally, new features were incorporated for *automatic sample extraction* and *flexible data querying*, enhancing the framework's original functionalities in the newly established **Expert-Musicomb**.

4.1. Automatic Sample Extraction

The automated sample extraction algorithm employs a maximization strategy to identify the longest uninterrupted sequences of silence in the composition after replacing all detected samples with empty elements. Given the function:

```
Function __inner__(sequence, min_ws, max_ws, initial_index):
    if $min\_ws > \lceil len(sequence)/2 \rceil$ then
        return [], 0
    else
        $subseq \leftarrow []$;
        for $ws$ in $min\_ws, \dots, max\_ws$ do
            $j \leftarrow initial\_index$;
            while $j < len(sequence) - ws$ do
                if $sequence[j : j + ws] == sequence[j + ws : j + 2ws]$ then
                    append $sequence[j : j + ws]$ to $subseq$;
                    $sequence[j : j + 2ws] \leftarrow \emptyset$;
                    $j \leftarrow j + 2ws$
                else
                    $j \leftarrow j + 1$
                end
            end
        end
        return subseq, silence_length
    end
end
```

where, *min_ws* and *max_ws* represent the size range of potential samples, and *initial_index* denotes the starting position for sliding windows within the sequence. By varying *initial_index*, different sample collections are generated based on their initial positions. The goal is to identify the collection of samples specified by *initial_index* such that the following statement is satisfied:

```
for $i$ in $0, \ldots, max\_ws$ do
    $subseq, max\_silence\_length \leftarrow$ __INNER__(samples, min_ws, max_ws, i);
    return $subseq$ if $max\_silence\_length == len(sequence) \vee max\_silence\_length$ is
      MAX;
end
```

Indeed, if the minimum and maximum sample sizes remain constant, as is typically the case since excessively long samples are avoided, the algorithm operates in linear time.

4.2. Dataset Enlargement

Expanding the dataset is crucial for advancing MusiComb. Integrating additional datasets into the framework enhances its capacity by augmenting the sample pool. This, in turn, broadens the spectrum of potential outcomes during generation. Moreover, it introduces fresh values for user-selectable parameters, such as a wider array of genres and an extended range of chord progressions. The dataset expansion employed two primary approaches:

- Incorporating additional MIDI datasets and extracting the required features using the methods previously described in Section 3.
- Creating multiple chord progressions for each melody sample by adjusting the initial conditions of the chord estimation model.

Introducing multiple chord progressions for the same melody sample offers a significant advantage. This approach expands the dataset without the need for additional samples. Simultaneously, it empowers the framework to merge the same melody sample with various chord progressions, fostering a higher level of music generation. Because of the recurrent nature of the chord progression estimation model, a slight alteration in the initial condition beyond the START token results in different outputs for the same input sample. To implement this effectively in the final approach, a set of varied initial conditions that guarantee **distinctiveness** and **harmonic consistency** in predictions needs to be identified. To ensure harmonic consistency, the initial conditions are chosen from a subset of possible conditions encountered by the model during training. Leveraging the RFDs Similarity discussed in Section 3 we can trust that the model has familiarity with these initial conditions. To reduce the likelihood of repetitions, a small subset, chosen as the nine most common initial conditions present in the training set, is selected.

sample	initial conditions[*]	estimated Chord Progression
commu00001.mid	[]	F-C-Dm-A#-F-C-Em-D
commu00001.mid	vi-IV-V	Dm-Am-Dm-A#-Gm-C-Am-D
commu00001.mid	V-vi-IV	Am-F-C-F-Am-C-Em-D
commu00001.mid	I-IV-V	Am-C-Dm-F-A#-C-Dm-D
commu00001.mid	V-IV-III	Am-G-Dm-F-A#-C-Dm-D
commu00001.mid	vi-iii-IV	F-Am-Em-F-F-Am-Em-D
commu00001.mid	ii-V-I	Am-Am-G-F-F-C-Em-D
commu00001.mid	I-VIIb-I	Dm-Am-A#-A#-Dm-Am-A#-D
commu00001.mid	IV-I-V	Am-F-C-F-Am-C-Em-D

[*] Initial conditions are represented in terms of Roman Literals

Table 6
Different predictions for Chord Progression over the same melody sample with different initial conditions. Highlighted in red is the only repeating element.

Table 6 presents an example of multiple chord progression estimations for the same MIDI sample together with the set of chosen initial conditions. This method enables the framework to effectively capture the relationship between a melody and potential chord progressions. Consequently, each melody sample can be paired with various chord progressions, enhancing the truthfulness of the overall music generation process without limiting each sample to a single chord progression.

4.3. Non-strict Data Query

In each generation cycle, once the user has specified their desired parameters, the framework seeks out samples that match the specified parameters and then employs a subset of these samples for music generation. Following dataset enlargement, the subsequent step to enhance the framework's capabilities involves loosening the criteria used for sample selection. The goal is to increase the number of selected samples for a given set of user-defined parameters (BPMs, genre, time signature, chord progression and key) while maintaining the harmonic and structural consistency ensured by strict selection rules. Due to the nature of the parameter, not all selection rules can be made less strict, but certain modifications are possible, details will be provided in the following subsections.

4.3.1. BPMs

The BPM parameter allows for a more flexible selection approach. Staying close to the desired BPM value, rather than fixing a specific one, adjustments can be made without significantly altering the sample's structure while enlarging the set of eligible samples. For a desired BPM represented by x, the sample's BPM by $bpm(n)$, and a selection indicator $choose(n)$, the selection rule becomes less strict to accommodate minor BPM variations by changing from (1) to (2).

$$choose(n) \iff bpm(n) = x \tag{1}$$

$$choose(n) \iff x - \alpha K \leq bpm(n) \leq x + \alpha K \tag{2}$$

where, K represents half of the maximum interval for the neighborhood, and α is a user-selectable parameter ranging from 0 to 1. Table 7 demonstrates that neighbourhood selection rules enable a

	Mean samples per BPM value
rigid selection	506
neighborhood selection	2204*

* neighbourhood selection parameters: $K = 20$ and $\alpha = 0.5$

Table 7
Mean number of samples for BPM values in the ComMU dataset with different selection rules.

broader range of samples to be included for the same BPM value. To prevent the neighbourhood interval from becoming excessively large and causing matching issues with the samples, the parameter K remains constant. Additionally, with the neighbourhood selection rule, BPM values that are not present in the dataset can be chosen as long as they fall within the interval of an existing BPM value. This marks another advancement compared to MusiComb, where only BPM values existing in the dataset were available for selection.

4.3.2. Harmonic Key

While the harmonic key is typically considered a less flexible parameter since Samples with different keys may not sound harmonious together, it is still possible to transpose a music piece to the desired harmonic key. We maintain consistency in the dataset by transposing all samples to the same harmonic key (*A minor/C major*) during the feature extraction process. The user can then freely choose the desired harmonic key. This ensures that most samples in the dataset can be selected, contingent on whether the user's chosen key is *minor* or *major*. Subsequently, a straightforward transposition to the desired key is performed after generating the final music piece.

Harmonic Key	Rigid selection (ComMU)	transposition (ComMU)
A minor	4729	4729
C major	6415	6415
other minor keys	/*	4729
other major keys	/*	6415

* original ComMU dataset only contains *A minor* and *C major* keys.

Table 8
Number of samples that can be chosen for a given harmonic key for both rigid selection and selection with transposition.

4.3.3. Chord Progression & Time Signature

We also study how chord progression and time signature can be treated less strictly. The incorporation of multiple chord progressions for each melody sample, as described Section 4.2, relaxes the selection rule for chord progressions. This expands the range of possible combinations, enabling a single melody to harmonize with various chord progressions. Moreover, while two samples with different time signatures cannot be seamlessly layered together, users can be allowed to choose between multiple time signature selections; this leads to music samples with time signature changes.

5. Conclusions

In our efforts to expand and enhance the existing MusiComb framework, we introduced a series of methodologies and techniques that are crucial for its future development. Specifically, we presented two models for *Track Role* and *Chord Progression* estimation, along with a set of new rules for the sample selection process, which significantly bolster the framework's capabilities, and an automatic sample extraction algorithm. Our work underscores the importance of data quality in modern Generative AI systems, demonstrating that it plays a pivotal role alongside the capabilities of the framework itself. Combinatorial systems, such as MusiComb, heavily depend on the quality of the information available about the involved elements. It's evident that even a state-of-the-art framework, when operating under incorrect preconditions (such as inaccurate sample information), may produce low-quality outputs. Conversely, datasets annotated by humans may be limited in size and require substantial time to expand. Striking a balance between data quality and the time needed to acquire it is therefore crucial for a continually growing and evolving framework like MusiComb. Our systems have demonstrated robust consistency with the data labels, serving not merely as tools for feature extraction but also introducing new degrees of freedom. Even by solely utilizing the original ComMU dataset, we can explore new and diverse generations while maintaining a high level of reliability in the quality of output.

Furthermore, there is ample opportunity for new introductions and future advancements. As previously mentioned, MusiComb is an ever-expanding framework capable of further expression beyond its current capabilities. Among the potential areas for research, significant developments may involve: (1) modifying the combinatorial backbone to enable the framework to integrate various chord progressions and time signatures within the same musical piece. (2) Utilizing the *Transformer XL*, as presented in [1], to generate missing samples or roles for specific generations. This approach allows the framework to seamlessly incorporate both dataset and generated samples within the same piece.

Acknowledgments

This work has been supported by the project TAILOR (funded by European Union's Horizon 2020 research and innovation programme, GA No. 952215).

References

[1] L. Hyun, T. Kim, H. Kang, M. Ki, H. Hwang, K. Park, S. Han, S. J. Kim, Commu: Dataset for combinatorial music generation, ArXiv abs/2211.09385 (2022).

[2] L. Giuliani, F. Ballerini, A. D. Filippo, A. Borghesi, Musicomb: a sample-based approach to music generation through constraints, in: 2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE Computer Society, Los Alamitos, CA, USA, 2023, pp. 194–198.

[3] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, D. Eck, Music transformer, 2018. arXiv:1809.04281.

[4] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, M. Sharifi, N. Zeghidour, C. Frank, Musiclm: Generating music from text, 2023. arXiv:2301.11325.

[5] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, I. Sutskever, Jukebox: A generative model for music, ArXiv abs/2005.00341 (2020).

[6] Q. Huang, D. S. Park, T. Wang, T. I. Denk, A. Ly, N. Chen, Z. Zhang, Z. Zhang, J. Yu, C. H. Frank, J. Engel, Q. V. Le, W. Chan, W. Han, Noise2music: Text-conditioned music generation with diffusion models, ArXiv abs/2302.03917 (2023).

[7] S. Dadman, B. A. Bremdal, B. Bang, R. Dalmo, Toward interactive music generation: A position paper, IEEE Access 10 (2022) 125679–125695.

[8] M. Allan, C. Williams, Harmonising chorales by probabilistic inference, in: L. Saul, Y. Weiss, L. Bottou (Eds.), Advances in Neural Information Processing Systems, volume 17, MIT Press, 2004.

[9] D. Cope, Computer modeling of musical intelligence in emi, Computer Music Journal 16 (1992) 69–83. URL: http://www.jstor.org/stable/3680717.

[10] S. F. Stanisław A. Raczyński, E. Vincent, Melody harmonization with interpolated probabilistic models, Journal of New Music Research 42 (2013) 223–235.

[11] J.-F. Paiement, D. Eck, S. Bengio, Probabilistic melodic harmonization, in: L. Lamontagne, M. Marchand (Eds.), Advances in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 218–229.

[12] D. Herremans, E. Chew, Morpheus: Generating structured music with constrained patterns and tension, IEEE Transactions on Affective Computing 10 (2019) 510–523.

[13] C. Anderson, A. Eigenfeldt, P. Pasquier, The generative electronic dance music algorithmic system (GEDMAS), Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment 9 (2021) 5–8. URL: https://doi.org/10.1609%2Faiide.v9i5.12649. doi:10.1609/aiide.v9i5.12649.

[14] F. Pachet, A. Papadopoulos, P. Roy, Sampling variations of sequences for structured music generation, in: International Society for Music Information Retrieval Conference, 2017.

[15] C. Nardi, Library music: technology, copyright and authorship, Current Issues in Music Research: Copyright, Power and Transnational Musical Processes. Lisboa: Edições Colibri (2012) 73–83.

[16] A. Zils, F. Pachet, Musical mosaicing, in: Digital Audio Effects (DAFx), volume 2, 2001, p. 135.

[17] C. Raffel, Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching, Columbia University, 2016.

[18] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, R. Bittner, The MUSDB18 corpus for music separation, 2017. URL: https://doi.org/10.5281/zenodo.1117372. doi:10.5281/zenodo.1117372.

[19] R. David, Mathematical harmony analysis (2016).

[20] L. Hyungui, R. Seungyeon, L. Kyogu, Chord generation from symbolic melody using blstm networks, 18th International Society for Music Information Retrieval Conference (ISMIR 2017) (2017).