Style-Specific Melody Generation

Gizem BALIKCI
(B.Sc., Izmir Institute of Technology, Turkiye, 2018)

A THESIS SUBMITTED FOR THE DEGREE OF
MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE

2026

Supervisor:

Associate Professor LEOW Wee Kheng

Examiners:

Associate Professor WANG Ye

Associate Professor Terence SIM

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

GIZEM BALIKCI
2026

ACKNOWLEDGEMENTS

I express my deepest gratitude to my supervisor Professor Leow Wee Kheng for his invaluable support and guidance throughout the course of this research. His encouragement and insightful feedback have been essential to the completion of this thesis.

I am also sincerely thankful to the National University of Singapore for giving me the opportunity to pursue and complete this work within its academic environment.

Furthermore, I would like to extend my sincere appreciation to my family, my mother Guler Balikci, my father Ercan Balikci, and my sister Irem Balikci, for their endless and unconditional love and support, even from far away across the world. Their encouragement, despite the long distance, has been a constant source of strength. Most importantly, I am truly grateful to my fiance Ali Koksal for always being by my side, offering unwavering support and companionship throughout this journey.

Abstract

Melody generation is a fundamental task in algorithmic music composition, as melody serves as the expressive core of musical works across genres and traditions. Style-specific melody generation aims to produce musically coherent sequences that embody distinctive stylistic characteristics. Existing approaches exhibit notable shortcomings. Rule-based methods require extensive manual definition for each style. Evolutionary algorithms rely on difficult-to-design fitness functions. Markov models lack capacity for long-term structural representation. Deep neural networks depend on large, style-specific training datasets. When trained on mixed-style data, such networks often generate melodies with diluted or confused stylistic identity.

Inspired by the creative practices of composers who reference existing music to guide stylistic decisions, this thesis proposes a method that generates melodies using a small amount of style-specific input, as little as a single piece. By combining probabilistic modeling with direct optimization, the approach eliminates the need for handcrafted rules or model training. Qualitative results demonstrate its ability to produce structured melodies that reflect the stylistic characteristics of the reference, offering a flexible and efficient solution across diverse musical styles.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Thesis Objective 2

2 Music Background **4**

2.1 Fundamental Musical Elements 4

2.2 Music Structure 5

3 Related Work **6**

3.1 Rule-Based Methods 6

3.2 Evolutionary Algorithms 7

3.3 Markov Models 9

3.4 Deep Neural Networks 10

3.5 Hybrid Methods 11

3.6 Summary 12

4 Style-Specific Melody Generation **14**

4.1 Melody Representation 14

4.2 Style Representation 15
4.3 Melody Generation 17

5 Experiments and Discussion **26**

5.1 Note Generation Tests 26
5.2 Pitch Generation Tests 31
5.3 Full-length Melody Generation 35

6 Conclusion **41**

References **43**

List of Tables

3.1 Comparison of existing methods. 13

List of Figures

4.1 Tree structures illustrating the best case, worst case, and typical case scenarios of the pitch generation algorithm. 23

5.1 Reference songs used to derive note models. (a) Pop-style reference song, *Over the Rainbow*. (b) Jazz-style reference song, *Fly Me to the Moon*. 28

5.2 Note generation from single reference. (a) Pop-style pitch sequence with pop-style note sequence. (b) Jazz-style pitch sequence with jazz-style note sequence. 29

5.3 Note generation from multiple references. (a) Pop-style pitch sequence with pop-style note sequence. (b) Jazz-style pitch sequence with jazz-style note sequence. 30

5.4 Pitch generation from single reference. (a) Pop-style pitch sequence with pop-style note sequence. (b) Jazz-style pitch sequence with pop-style note sequence. (c) Jazz-style pitch sequence with jazz-style note sequence. 32

5.5 Pitch generation from multiple references. (a) Pop-style pitch sequence with pop-style note sequence. (b) Jazz-style pitch sequence with pop-style note sequence. (c) Jazz-style pitch sequence with jazz-style note sequence. 34

5.6 Examples from exhaustive generation of pitch sequences. (a) Pitch sequence that matches the reference song *Beauty and the Beast* exactly. (b), (c) Variations of pitch sequences. 35

5.7 Section A of the reference song, *I Want to Hold Your Hand*, and sections in the style of section A of the reference song, generated by different methods: (a) Section A of the reference song. (b) Result of the proposed method. (c) Result of Magenta Lookback-RNN. (d) Result of Magenta AttentionRNN. 37

5.8 Section B of the reference song, *I Want to Hold Your Hand*, and sections in the style of Section B of the reference song, generated by different methods: (a) Section B of the reference song. (b) Result of the proposed method. (c) Result of Magenta Lookback-RNN. (d) Result of Magenta AttentionRNN. 38

5.9 Pitch style reference song, *Love Me Tender*, and sections generated by the proposed method: (a) Section A of the pitch style reference song. (b) Result of the proposed method generated according to reference section A. (c) Section B of the pitch style reference song. (d) Result of the proposed method generated according to reference section B. 40

Chapter 1

Introduction

1.1 Motivation

Automated music generation refers to the algorithmic generation of music, typically focusing on two main tasks: melody generation and harmonization. Among these, melody generation is fundamental, as the melody forms the core of a musical piece and is often its most distinctive and expressive element. Other musical components, such as harmony, primarily serve to support and enhance the delivery of the melody.

Melody generation involves producing a sequence of musical notes that form a musically pleasing and coherent structure. Melody comes in various flavors or styles, which can be described at multiple levels. At the coarse level, melodic style may align with a genre, such as pop, jazz, or classical, or a cultural tradition, such as Western, Chinese, or Indian. It may also reflect a historical period, such as Baroque, Classical, or Romantic, or the distinctive approach of an individual composer. At the fine level, different melodies may convey different stylistic identities. More generally, melodic style can be inferred from consistent characteristics observed across one or more musical pieces.

Existing melody generation methods can be grouped into several categories: rule-based approaches, evolutionary algorithms, Markov models, and deep neural networks. Rule-based methods produce melodies by applying manually defined rules that guide note progression, often grounded in music theory to maintain coherence. Evolutionary algorithms iteratively refine a population of candidate

melodies through a process of evaluation and selection based on fitness functions that assess musical quality. The core idea behind Markov models is probabilistic modeling of note-to-note transitions, using statistics derived from existing musical examples, to generate melodies that resemble characteristic aspects of the source material. Deep neural networks learn complex, non-linear relationships from large datasets and generate melodies that capture patterns found in the training examples.

Existing melody generation methods have several shortcomings. Rule-based systems require handcrafted rules. It is challenging to define a rule set that effectively handles a musical style. Each style demands a distinct set of rules, making it difficult for such systems to manage stylistic variation. Handling different styles involves manually defining new rules, which is a tedious process that complicates adaptation to new styles. Evolutionary algorithms rely on style-specific fitness functions, which are difficult to design for complex or loosely defined musical styles. Such functions require explicit, quantifiable criteria that are difficult to define, limiting their applicability across diverse stylistic domains. Markov models capture local stylistic patterns by modeling note-to-note transitions based on short-term statistical dependencies. It is very difficult for them to represent long-term musical structure because their memory is limited to a fixed number of preceding states. Thus, it is difficult for them to encode broader thematic development, phrase-level continuity, or hierarchical relationships that span larger musical segments. Deep neural networks are widely used in recent years because they can learn from training examples to generate melodies. Although these methods do not require manual crafting of rules and fitness functions, they still require manual preparation of large amounts of training examples. Moreover, the style of the generated melodies depend directly on the training examples. To generate melodies of different styles, a DNN has to be trained separately on different training sets for different styles. If it is trained on a training set with mixed style, it will mix the style elements and generate melodies with confused or diluted style identity.

1.2 Thesis Objective

Modern song composers often seek stylistic inspiration by listening to reference music that reflects the aesthetic qualities they aim to reflect in their own work. This process may involve adopting stylistic elements directly or adapting them to align with a specific stylistic direction. Inspired by this artistic process, this

thesis aims to develop a method for melody generation based on a small amount of style-specific reference music provided by the user.

The proposed approach uses probabilistic models to capture note sequence characteristic of the reference style, given as few as a single reference piece. It applies direct optimization method, removing the need for model training or large annotated datasets. The method generates melodies that reflect the desired music structure, with clearly defined phrase and section closure.

The contributions of this thesis can be summarized as follows:

- **Style-specific melody generation:** The proposed method enables the generation of melodies that reflect the stylistic characteristics of a user-provided reference, which can be as few as a single music piece. This supports personalized and context-aware composition.
- **Flexible adaptation to diverse musical styles:** Unlike methods that require retraining or rule redefinition for each style, the proposed approach can adapt to different stylistic inputs without manual rule crafting, model training or extensive data preparation, making it suitable for a wide range of genres and traditions.
- **Training-free direct optimization:** The method employs probabilistic modeling and direct optimization techniques, eliminating the need for large annotated datasets or model training. This significantly reduces computational overhead and simplifies the generation pipeline.

Chapter 2

Music Background

2.1 Fundamental Musical Elements

The most fundamental element of musical structure is the note, defined by three attributes: pitch, duration, and onset. Pitch refers to the perceived frequency of a sound, determining whether a note is heard as high or low. It is a perceptual property determined by both the physical vibration of the sound and the listener's perception. Duration indicates the relative length of a note, commonly expressed as fractional values such as 1 (whole note), 1/2 (half note), 1/4 (quarter note), 1/8 (eighth note), and finer subdivisions. Onset is the time point at which a note begins within a melody. A rest represents the absence of pitch but, like a note, it is defined by duration and onset, making it possible to place and measure silence just like a note.

The temporal structure of music is organized into segments called bars (or measures), each containing a specific number of beats, which serve as basic units of time. The grouping and duration of these beats are defined by the time signature, a notational convention written as a fraction, e.g., 2/4, 3/4, 4/4, or 6/8. In this fraction, the upper number indicates how many beats occur in each bar, and the lower number specifies the note value assigned to one beat. For instance, the time signature 4/4 indicates that each bar contains four beats, with each beat corresponding to a quarter note.

Pitches are organized through tonal relationships, one of the most fundamental being the octave, defined by a doubling of frequency. Within each octave, pitches

are grouped into 12 pitch classes, namely C, C♯ or D♭, D, D♯ or E♭, E, F, F♯ or G♭, G, G♯ or A♭, A, A♯ or B♭ and B. Notes assigned to the same pitch class are perceived as closely related in tonal identity. The smallest interval between pitches is the semitone, which is the difference between adjacent pitch classes, for example, C and C♯, or E and F.

2.2 Music Structure

A piece of music is organized into sections, phrases and bars. The number of bars in a phrase is typically a multiple of 4, such as 4, 8, 16, 32, etc. The number of phrases in a section can be 1, 2, 4, 8, 16, etc.

There are many kinds of section organizations such as A B A, A B A C A, etc. In the style of A B A, section A is repeated twice, and section B is composed to contrast section A. Therefore, the melody of section B is coherent but contrast with that of section A. A variation of A B A is A B A', where section A' is essentially the same as section A, except for minor variations. Similarly, the phrases in a section may be repeated exactly or repeated with minor variations. It is uncommon for a long section to contain several unique, contrasting phrases because it loses internal coherence.

Chapter 3

Related Work

Melody generation has been explored through a variety of computational methods, each employing distinct techniques to generate melodies. This chapter reviews the categories of existing methods, including rule-based methods, evolutionary algorithms, Markov models, and deep neural networks. Additionally, hybrid methods that combine elements from multiple paradigms are discussed, highlighting efforts to leverage complementary strengths and address inherent limitations of individual approaches. Each subsection provides a focused review of representative techniques, their underlying principles, and their contributions to the field of algorithmic melody generation.

3.1 Rule-Based Methods

Rule-based methods generate melodies by applying symbolic rules that guide how pitches, onsets and durations are chosen, grouped into patterns, and arranged over time. The rules are constructed either manually, using domain knowledge in music composition, or automatically, by extracting patterns from music examples through grammar-based or probabilistic techniques. During melody generation, these rules are applied to produce note sequences by specifying which pitches, pitch onsets and durations are selected, how they are combined into coherent phrase structures, and how they are arranged within the overall temporal structure of the composition.

Rule-based methods have been applied to generate melody in the style of com-

posers, including Bach, Mozart and Milton Babbit [1–6], to generate contrapuntal melodies [7–10], and to model rhythmic patterns [11, 12].

Rule-based systems can incorporate deterministic, fuzzy, or probabilistic rules. Deterministic rules function as hard constraints, formulated as Boolean functions that must evaluate to true for a rule to be satisfied. Fuzzy rules allow partial satisfaction by assigning degrees of truth, acting as soft constraints that support less strict rules [11, 12]. Probabilistic rules introduce variability by assigning probabilities to rule applications, enabling controlled randomness in the composition process [2, 5, 7–10, 13].

Rule-based methods have several strengths in algorithmic composition. They are structurally simple and computationally efficient as they follow explicitly defined rules rather than relying on complex calculations or iterative processes. Rule-based methods are training-free, which makes them flexible and easy to adapt. They can generate melodies with clear structural organization through rules that explicitly encode structure such as phrase boundaries. They can be modified or extended by adding new rules without requiring retraining.

Rule-based methods have several shortcomings in algorithmic composition. They require significant domain knowledge, which makes the process of defining effective rules tedious and challenging. They cannot easily capture patterns that apply across all melodies, as nuances in musical composition make it difficult to design rules that remain effective in varied musical contexts. They must rely on separate rule sets to generate different musical styles, and creating these sets is a tedious and challenging process that makes it difficult for rule-based systems to handle diverse styles effectively.

3.2 Evolutionary Algorithms

Evolutionary algorithms generate melodies by gradually improving a population of music examples through a repeated process of evolution, selection, and reproduction. They begin with an initial set of candidate melodies, either generated randomly or sampled from a reference set. Evolutionary algorithms evaluate each candidate using a fitness function and select a subset of high-performing candidates as parents for the next generation. They produce new candidates using variation operators. One common operator is crossover, which generates a new melody by combining fragments from two parent melodies. Another is mutation, which randomly alters elements of a parent melody to produce a new one. These opera-

tions are applied iteratively to generate new candidates and refine the population over successive generations.

Evolutionary algorithms have been used to generate rhythmic patterns [14]. They have also been used to generate melodies with balanced smoothness and tension [15], where tension refers to musical dissonance. In addition, they have been applied to generating short melodic transitions that connect two fragments of melody [16]. These methods have also been used to generate melodies conditioned on chord sequences [17–19] and to produce both melody and accompaniment [20]. In style-specific settings, they have been used to generate four-voice melodies in the style of Bach [21–23] and traditional church hymns [24].

Evolutionary algorithms for melody generation vary according to how the fitness function is defined. One way of defining the fitness function is as a combination of multiple musical heuristics through weighted sums or multi-objective optimization [17–20, 22, 24–27]. It is also defined as a measure of melodic similarity, typically focusing on pattern similarity [16, 28–30]. Alternatively, it is defined as statistical features derived from a reference set, including pitch distributions or n-gram frequencies [14, 15, 21, 23, 31].

The strength of evolutionary methods is that they do not require training. Instead, they generate melodies through iterative refinement. By focusing on the most promising candidates, they generate music that satisfies predefined quality standards set by the fitness function.

One of the shortcomings of evolutionary methods is their computational cost. These algorithms rely on evolving large populations over many iterations, which can result in high resource demand. Evolutionary methods also have limited structure awareness due to the difficulty of defining fitness functions for global structure, even though they can sometimes capture medium-level features. Another shortcoming is their limited adaptability to specific musical styles. Defining style-specific fitness functions is challenging, as it requires formalizing nuanced stylistic features with sufficient specificity. Without clearly defined criteria, these methods cannot reliably generate outputs that reflect distinct stylistic characteristics

3.3 Markov Models

Markov models generate melodies by modeling the probability of transitioning from one state to another, such as from one pitch to the next. These models assume that the next state in a sequence depends only on a limited number of previous states. A first-order Markov model considers only the current state, while higher-order models take into account several previous states. By learning these transition probabilities from the reference set, Markov models can produce new melodies that reflect the statistical patterns of the reference set.

In melody generation, two variations of Markov models have been used, namely, Markov chains [32–43] and Hidden Markov Models (HMMs) [44–46]. Markov chains consider musical states, such as pitch, note duration, or pitch-duration pairs, as fully observable states and compute state transition probabilities directly from sample data. These probabilities are embedded into a state transition matrix, and guide sequence generation using algorithms such as random walk, selecting each state based on its predecessor.

On the other hand, HMMs extend Markov models through the inclusion of hidden states. These models compute transition probabilities between hidden states and emission probabilities, which represent the likelihood of a particular observable state arising from a given hidden state. Melody generation with HMMs typically involves techniques such as dynamic programming and maximum likelihood estimation.

Markov models are effective for melody generation due to their ability to capture statistical patterns in musical sequences. Markov chains, in particular, are computationally efficient and easy to implement, making them well suited for modeling short-term musical structures such as relationships between neighboring notes. HMMs extend the capabilities of basic Markov chains by incorporating hidden states, allowing the modeling of long-range dependencies across multiple preceding states.

Despite the strengths of Markov models, they also have notable limitations. One of their main drawbacks is that both low-order and high-order Markov chains often lack global structural awareness unless incorporated with constraints or rule-based methods. Although higher-order Markov chains can model longer-range dependencies, this increased representational capacity often requires greater computational complexity and carries a higher risk of sparsity in the transition matrix. Moreover, while transition probabilities can be learned from a single data sample, this also introduces a risk of sparsity that increases with the complexity of the

model. Finally, HMMs incorporate additional underlying information, but this makes them more complex and less efficient than Markov chains.

3.4 Deep Neural Networks

Deep neural networks (DNNs) generate melodies by learning complex, nonlinear mappings between numerical representations of inputs and outputs. During training, the network modifies its internal weights through stochastic gradient descent and backpropagation to capture statistical patterns in the training data. In the generation phase, it produces output via feedforward activation that reflects the characteristics captured during training.

The types of DNNs used in melody generation include Recurrent Neural Networks (RNNs) [47–51], Transformers [52, 53], Variational Autoencoders (VAEs) [54–56], and Generative Adversarial Networks (GANs) [57]. RNNs are neural architectures designed to model temporal relationships in sequential data. They generate one element at a time, with each prediction influenced by earlier steps through cyclical neuron connections that transmit information across time. An extension of RNNs, Long Short-Term Memory (LSTM) networks preserve information over longer time spans within a sequence.

Transformers offer an alternative approach to modeling sequence data. Instead of stepwise computation, they use a mechanism called self-attention to capture dependencies between all elements simultaneously. This design allows a long sequence to be processed in parallel rather than step by step.

VAEs are generative models trained to reconstruct input data. They consist of two components: an encoder and a decoder. The encoder maps input into an abstract space, referred to as the latent space. The decoder then reconstructs the original data from the latent representation. Training the VAE to minimize Kullback-Leibler divergence ensures that the model learns a meaningful mapping rather than replicating the input directly.

GANs are neural models trained to learn and sample from the probability distribution of training data. They consist of two components: a generator and a discriminator. The generator learns to map random samples from a standard distribution into data examples that resemble the training set, while the discriminator learns to distinguish these generated examples from those belonging to the training data. Training continues in an adversarial manner until both models reach

equilibrium, where the discriminator can reliably differentiate training data from generated data, and the generator produces outputs that the discriminator can no longer distinguish from training examples.

Neural networks are powerful models for melody generation due to their ability to learn complex, non-linear relationships from large datasets. They are particularly effective when trained with stylistically consistent training data, where they can learn recurring patterns of pitch and rhythm that characterize a given style. DNNs such as RNNs, transformers, VAEs, and GANs have been applied to melody generation tasks, each contributing unique strengths in sequence modeling and generative diversity.

However, neural networks also come with significant shortcomings. They require large amounts of training data, which can be a barrier when working with niche or underrepresented musical styles. Moreover, their outputs are highly dependent on the training set. Therefore, when trained on mixed-style data, they may produce outputs with mixed stylistic identity. Additionally, while they are effective at modeling short-term dependencies such as local transitions, they often struggle to capture long-range musical structure, such as phrase-level continuity or thematic development. This limitation is due to their dependence on immediate context, making it difficult to capture long-term structure or generate coherent phrases without explicit guidance by structural constraints.

3.5 Hybrid Methods

Hybrid methods combine elements from multiple melody generation approaches to leverage their complementary strengths and mitigate individual limitations. These methods often integrate symbolic rule-based systems and deep learning architectures to produce more musically coherent and stylistically flexible output.

For example, MeloForm [58] integrates an expert system with a neural network to generate melodies with explicit musical form. The expert system encodes domain knowledge through symbolic rules that define structural constraints, such as phrase boundaries and motif repetition. On the other hand, the neural network generates melodic content within these constraints and learns expressive patterns from training data. This approach addresses the limitations of purely rule-based systems in handling stylistic nuances and purely data-driven models in applying formal structure, enabling the generation of melodies that are both stylistically rich and structurally consistent.

Dai et al. [59] introduces another hybrid method that emphasizes controllability and structural coherence through a hierarchical music structure representation. The method integrates a deep neural network with a multi-level abstraction of musical form, enabling the generation of melodies that reflect both local musical details and global structural organization. By explicitly modeling motifs, phrases, and sections, the method allows users to guide the generation process with high-level structural constraints, addressing a common limitation of purely data-driven models that often lack formal coherence. This approach enhances stylistic control and musical expressiveness, making it suitable for applications where both creativity and structure are essential. The work exemplifies how hybrid systems can combine the adaptability of neural networks with symbolic representations to produce musically rich and structurally sound compositions.

These hybrid strategies exemplify a growing trend in algorithmic composition: combining interpretable, rule-based logic with the adaptive power of machine learning. In doing so, they offer promising solutions to challenges such as style generalization, long-term structure modeling, and musical expressiveness.

Both [58] and [59] rely on deep neural networks that require training. This process demands significant computational resources as well as tedious data preparation. In addition, adapting hybrid methods to different musical styles is challenging. Rule-based methods require creating separate rule sets to generate melodies of different styles, and neural networks require retraining for different styles. Consequently, it is difficult for them to achieve stylistic flexibility.

3.6 Summary

Existing melody generation methods each has its own strengths and weaknesses. Rule-based systems offer interpretability and efficiency, and they can explicitly encode musical structure such as phrase boundaries, but they struggle with stylistic flexibility. Evolutionary algorithms provide adaptive refinement through fitness-based selection, but they face challenges in defining style-specific and structure-aware evaluation criteria. Markov models excel at capturing local statistical patterns, but they lack awareness of long-term musical structure. Deep neural networks have demonstrated powerful pattern-learning capabilities, especially in style-specific contexts, though they require large training datasets and often have limited structural awareness. Hybrid methods attempt to bridge these gaps by combining symbolic, statistical, and neural components to enhance both expressiveness and structural coherence. Together, these approaches form a rich foundation for

Table 3.1: Comparison of existing methods.

Methods	References	Training Required	Computational Resource Requirement	Stylistic Flexibility	Structure Awareness
Rule-based	[1–13]	No	Low	Difficult	Yes
Evolutionary	[14–31]	No	High	Difficult	Limited
Markov Chains	[32–43]	No	Low	Easy	No
HMMs	[44–46]	Yes	High	Tedious	Limited
DNNs	[47–57]	Yes	High	Tedious	Limited
Proposed Method	NA	No	Low	Easy	Yes

algorithmic composition, highlighting the need for models that balance musical structure, stylistic control, and generative flexibility.

Table 3.1 provides a comparative summary of melody generation methods, evaluated along four dimensions: training requirements, computational resource requirement, stylistic flexibility, and structure awareness. The table also includes the proposed method introduced in this thesis, which aims to combine the simplicity and flexibility of statistical models with structural awareness without requiring training. This comparison highlights the strengths and weaknesses of existing methods and the need to balance computational complexity, stylistic control, and generative flexibility. These are the goals that motivate the methodology proposed in this thesis.

Chapter 4

Style-Specific Melody Generation

This chapter presents the proposed method for style-specific melody generation that produce musically coherent melodies. The method is structured into three main components: melody representation, style modeling, and melody generation. Melody representation defines the structural and temporal framework of the music, including notes, bars, phrases, and sections. Style modeling captures rhythmic and pitch characteristics from reference music using probabilistic models. Melody generation applies these models to produce new melodies, guided by musical constraints such as closure and acceptance condition. Together, these components form a training-free, flexible system for generating stylistically consistent melodies.

4.1 Melody Representation

A melody consists of a sequence of m bars h_i, for $i = 0, \ldots, m-1$. It has a time signature of 2/4, 3/4 or 4/4. The time signature of 6/8 is omitted for ease of explanation. Music representation and modeling can be easily extended to 6/8 music.

Each h_i corresponds to a single bar of music. Consequently, the length L of h_i is equal to the bar length, which is 2, 3, or 4 beats for time signatures of 2/4, 3/4, or 4/4, respectively. The bar length L is measured in beats, while the unit interval θ represents the smallest beat subdivision (e.g., 1/4). Given this structure, L is an integer multiple of θ, and the total number of discrete time steps within a bar is

L/θ.

A bar h_i consists of a sequence of n_i notes u_{ij} with note index $j = 0, \ldots, n_i - 1$. A note u_{ij} has three attributes, namely pitch p_{ij}, duration d_{ij} and onset time τ_{ij}. Pitch p_{ij} is encoded as a MIDI pitch number that ranges from 0 to 127. Duration d_{ij} is an integer multiple of θ. The onset time τ_{ij} is an integer ranging from 0 to $L/\theta - 1$, indicating the time index at which the note begins within a bar. If a note u_{ij} has onset time $\tau_{ij} = t$ and duration d_{ij}, it spans the time indices from t to $t + d_{ij}/\theta - 1$ within the bar. Time intervals within a bar that are not occupied by any note represent rests.

Bars are organized into phrases and sections. A phrase consists of 4, 8 or 16 bars, and a section consists of 1, 2, 4 or 8 phrases. Therefore, the number of bars m is related to the number of sections and section lengths. Bars h_i are assigned structure labels λ_i. The value of λ_i is 'P' or 'S' for bar h_i located at the end of a phrase or a section, and '', otherwise. If bar h_i repeats an earlier bar $h_{i'}$ exactly, then $\lambda_i = i'$. These information constitute the music structure of the melody.

Strong Beat

Let γ denote the number of equally spaced strong beats per bar. Specifically, $\gamma = 2$ for the 4/4 time signature, while $\gamma = 1$ for the 2/4 and 3/4 time signatures. Let t represent the discrete time step within a bar. A time step t corresponds to a strong beat position if and only if it satisfies the following condition.

$$t \bmod L/\gamma\theta = 0. \tag{4.1}$$

4.2 Style Representation

The style of melody consists of two aspects: rhythmic style and pitch style. The rhythmic style is represented by the **note model**. It captures the probability $P(t,d)$ that a note of duration d occurs at onset time t within a bar. The time index t ranges from 0 to $L/\theta - 1$. The note duration d is an integer multiple of θ. In addition, the note model captures the most frequently occurring note duration, denoted by d^*. Furthermore, the note model captures the time point t_l that denote the latest onset time at which a long note with duration $d > d^*$ occurs. It also captures ε, the probability that a bar begins with a long note of duration $d > d^*$.

The pitch style is represented by the **pitch transition model**. It captures the style of reference music in terms of the change in pitch from one note to the next. Notice that a melody sounds the same regardless of the actual octave of the pitches. Thus, it is better to capture pitch transition in terms of pitch classes than the actual pitch numbers. The pitch transition model therefore captures the transition probabilities between pitch classes. A pitch class is a set of pitches that are separated by an integer number of octaves. For instance, the pitch class C includes all pitches labeled as C across octaves, such as C0, C1, C2, and so forth. Middle C (C4) corresponds to MIDI pitch number 60, while pitch number 0 corresponds to C-1. Formally, the pitch class c associated with a given pitch number p is defined by:

$$c = p \bmod 12. \tag{4.2}$$

Let p and p' denote consecutive pitches in a reference music, where p' immediately follows p. Then, the pitch transition is represented as (c, c', δ), where c and c' are the pitch classes of p and p', respectively, and δ is the direction of transition:

$$\delta = \begin{cases} +1 & \text{if } p' > p \\ 0 & \text{if } p' = p \\ -1 & \text{if } p' < p \end{cases} \tag{4.3}$$

Then, the pitch transition model captures the transition probability $P(c, c', \delta)$.

Given a pitch transition (c, c', δ) and a pitch p in pitch class c, the pitch p' in pitch class c' can be obtained by a pitch mapping function $p' = f(p; c, c', \delta)$:

- $\delta = 0$: $p' = p$.
- $\delta = 1$, $c' > c$: $p' = p + (c' - c)$.
- $\delta = 1$, $c' < c$: $p' = p + (c' + 12 - c)$.
- $\delta = 1$, $c' = c$: $p' = p + 12$.
- $\delta = -1$, $c' < c$: $p' = p + (c' - c)$.
- $\delta = -1$, $c' > c$: $p' = p + (c' - 12 - c)$.
- $\delta = -1$, $c' = c$: $p' = p - 12$.

4.3 Melody Generation

4.3.1 Overview

The melody to be generated should follow a user-specified style. The easiest way for a user to specify the desired style is to provide one or more reference music. If more than one reference music is used, they should have consistent styles. Given the reference music, a pre-processor is used to extract the pitch transition model and note model. In addition, the user can provide information such as the key signature, time signature, and music structure, including section structure, phrase structure, and structure labels.

The method for generating style-specific melody consists of three parts, namely note generation, pitch generation and melody generation. The note generation algorithm determines the onset times and durations of the notes in a single bar. It is guided by the probabilistic note model and the specified musical structure. The pitch generation algorithm assigns pitches to the generated note sequence according to the pitch transition model and structural constraints. Finally, the melody generation algorithm applies the note and pitch generation algorithm iteratively to generate the melody one bar at a time, following the temporal and tonal characteristics of the reference music and conforming to the global structural requirements such as repetition and closure.

4.3.2 Conditions for Generating Melody

To make the generated melody sound musical, the generation of note sequence and pitch sequence should follow three conditions: acceptance condition, phrasing and closure.

1. Acceptance Condition

A sequence of repeated pitches can sound monotonous. A bar is considered monotonous if a sequence of α pitches appears exactly and consecutively for more than β times. The appropriate values of α and β depend on the desired music style, and are thus provided by the user.

2. Phrasing

In a well-formed melody, rhythmic organization should support phrasing, which in music theory refers to how listeners perceive melody parts as separate bars. Phrasing shapes the musical flow, allowing each bar to be understood as a separate idea. A common principle is the use of longer note values to mark boundaries. A sustained note at the beginning of a bar provides stability, while a long note at the end highlights the bar as a distinct segment. When both positions are emphasized, the bar is more readily perceived as a separate part of the phrase.

In this thesis, phrasing is defined as the presence of a long note at the beginning of a bar, at the end of a bar, or at both positions. A long note is formally defined as a note with duration $d > d^*$, where d^* denotes the most frequently occurring note duration.

3. Closure

In a well-formed melody, each phrase and section should end appropriately to indicate closure. In typical musical practice, the closure note must occur on a strong beat and have a duration that is sufficiently long.

In this thesis, closure is defined as the presence of a final note (t,d) such that the onset time t occurs on a strong beat and the duration d is sufficiently long, specifically $d > d^*$.

Additionally, the final pitch of a phrase often belongs to the tonic triad which is a group of three pitch classes derived from the first, third, and fifth notes of a scale. For example, in C major, the tonic triad consists of the notes C (tonic), E (third), and G (fifth). Ending on one of these notes reinforces tonal resolution. In some instances, however, a phrase may conclude on a pitch outside the tonic triad, resulting in a more open-ended conclusion while still maintaining a coherent sense of structural closure. Sections, by contrast, typically end with the tonic note, providing a clearer sense of finality and tonal closure.

4.3.3 Note Generation

The note generation algorithm generates a sequence of notes in a bar of length L and unit interval θ. The note sequence contains notes at onset time τ_j and duration d_j, for $j = 0, \cdots, n-1$. The notes are generated according to the structure label λ of the bar and the note model. Recall that the note model contains pairs (t,d), the

most frequently occuring duration d^*, the latest time point t_l at which a long note with duration $d > d^*$ occurs, and ε, the probability that a bar begins with a long note of duration $d > d^*$.

Algorithm 1 Note Generation Algorithm

Input: bar lenght L, unit interval θ, structure label λ, note model.
Output: τ_j, d_j, where $j = 0, \ldots, n-1$.

1: Initialize $j = 0$, $t = 0$, remaining duration $\omega = L$.
2: **repeat**
3: Generate random number ρ sampled from $[0, 1]$.
4: **if** $j = 0$ and $\rho < \varepsilon$ and there is a (t, d) with $d > d^*$ **then**
5: Select (t, d), break tie randomly.
6: **else if** ($\lambda =$ 'P' or 'S') and there is a (t, d) with t at a strong beat and $d > d^*$ **then**
7: Select (t, d), break tie randomly.
8: Set $\omega = 0$.
9: **else if** ($\lambda \neq$ 'P' or 'S') and there is a (t, d) with $t = t_l$ and $d > d^*$ **then**
10: Select (t, d), break tie randomly.
11: Set $\omega = 0$.
12: **else if** there is a (t, d) with $d \leq \omega$ **then**
13: Select (t, d), break tie randomly.
14: **if** a (t, d) is selected **then**
15: Create note u_j with onset time $\tau_j = t$ and duration $d_j = d$.
16: Advance note index: $j = j + 1$.
17: Advance time index: $t = t + d/\theta$.
18: Reduce remaining duration: $\omega = \max(0, \omega - d)$.
19: **else**
20: Skip time index, i.e., insert a rest: $t = t + 1$.
21: Reduce remaining duration $\omega = \omega - \theta$.
22: **until** $\omega = 0$

The note generation algorithm works as follows:

- Lines 4 and 5 select (t, d) for beginning long note of a bar, if such a (t, d) exists in the note model. The condition $\rho < \varepsilon$ governs the probability of setting a long note at the beginning of a bar. The parameter ε is the probability that a long note occurs at the beginning of a bar in the note model. Then, the condition of $\rho < \varepsilon$, with $0 <= \rho <= 1$, has the same probability

as ε. Therefore, the generated note sequence follows the same probability as the note model for having a long note at the beginning of a bar.

- Lines 6, 7 and 8 select (t,d) for ending long note that satisfies the closure condition. A bar with structure label 'P' or 'S' is located at the end of a phrase or a section. For this bar, selecting a long note at a strong beat t ensures closure condition. This (t,d) exists for the last note of the reference music. Thus, it is in the note model and can always be selected. Line 8 ensures that the selected note becomes the last note in the bar. The remaining duration is set to zero so the algorithm will not generate any further notes.

- Lines 9, 10 and 11 select (t,d) for ending long note that satisfies the phrasing condition. The phrasing condition is applied to a bar that is not at the end of a music phrase or a section. It requires the bar to end with a long note. The latest onset time t_l is the position where such a note can be placed. Thus, selecting a long note with onset $t = t_l$ leads to the phrasing condition being satisfied. Line 11 ensures that the selected note becomes the last note, similar to Line 8.

- Lines 12 and 13 select (t,d) for other notes.

- Lines 15 to 18 create a note, advance the note index and time index, and reduce remaining duration.

- Lines 20 and 21 advance time index and reduce remaining duration without creating a note. In other words, it creates a rest.

Algorithm 1 repeats until the remaining duration is zero. Note that the conditions in Line 4 and Line 9 may be both satisfied at different time steps. Thus, Algorithm 1 may generate a long note both at the beginning and the end of a bar. On the other hand, the conditions in Line 6 and Line 9 cannot be both satisfied for the same bar. Thus, Algorithm 1 generates a long ending note that satisfies either closure or phrasing condition but not both.

Algorithm 1 repeats for $n < L/\theta$ iterations. Each iteration creates a note by selecting an appropriate (t,d). The number of (t,d) pairs in the note model is a constant that is independent of n. Therefore, the time complexity of the algorithm is $O(n)$.

4.3.4 Pitch Generation

The pitch generation algorithm generates k candidate pitch sequences for a bar. Instead of generating just one sequence, it generates k sequences for the melody generation algorithm to choose. It generates a pitch sequence according to the note sequence generated by the note generation algorithm and an initial pitch, which is generated by the melody generation algorithm. The pitches in a bar are generated based on the pitch transition model and the structure label λ of the bar. Recall that the pitch transition model contains transition triplets (c, c', δ) and their transition probabilities.

Algorithm 2 Pitch Generation Algorithm

Input: Note sequence, structure label λ, starting pitch p_0, melody key, pitch transition model.

Output: k candidate pitch sequences.

1: Create an empty stack S.
2: Initialize threshold $0 < \phi < 1$.
3: Create root node q_0 with pitch p_0 and depth 0.
4: Push node q_0 onto stack S.
5: **repeat**
6: Pop a node from stack S and set it as the current node q.
7: **if** depth of $q = n - 1$ **then**
8: Record pitches from q_0 to q as a candidate pitch sequence.
9: **else if** depth of $q = n - 2$ and ($\lambda =$'S' or 'P') and random number ρ sampled from $[0, 1] > \phi$ **then**
10: Select all 3 pitches p' of the tonic triad of the melody key.
11: **else**
12: Select M pitches p' with largest transition probability from pitch of q.
13: **for each** p' in increasing transition probability **do**
14: **if** pitch sequence from p_0 to p' is acceptable **then**
15: Create a node q' as a child node of q.
16: Set pitch of q' as p', depth of q' as depth of $q + 1$.
17: Push node q' onto stack S.
18: **until** k candidate pitch sequences are generated

In generating the pitch sequences, the pitch generation algorithm employs a tree structure to maintain partial pitch sequences. It works as follows:

- Lines 1 to 4 create a root node q_0 on the stack with initial pitch p_0 and depth

of 0.

- Line 6 pops the node q on the top of the stack.
- Lines 7, 8: If depth of q is $n-1$, q is the last node in the sequence. So, the pitch sequence from root node q_0 to node q is recorded as a candidate pitch sequence.
- Lines 9, 10: If depth of q is $n-2$ and the bar is at the end of a phrase or a section, then 3 pitches of the tonic triad of the melody key are selected probabilistically as candidate next pitches p'.
- Lines 11, 12: Most of the time, M candidate pitches p' with the largest transition probabilities from the pitch of q are selected.
- Lines 13 to 17 checks whether the new pitch p' forms an acceptable sequence from the initial pitch p_0. If true, a node q' with pitch p' is created as the child node of q.

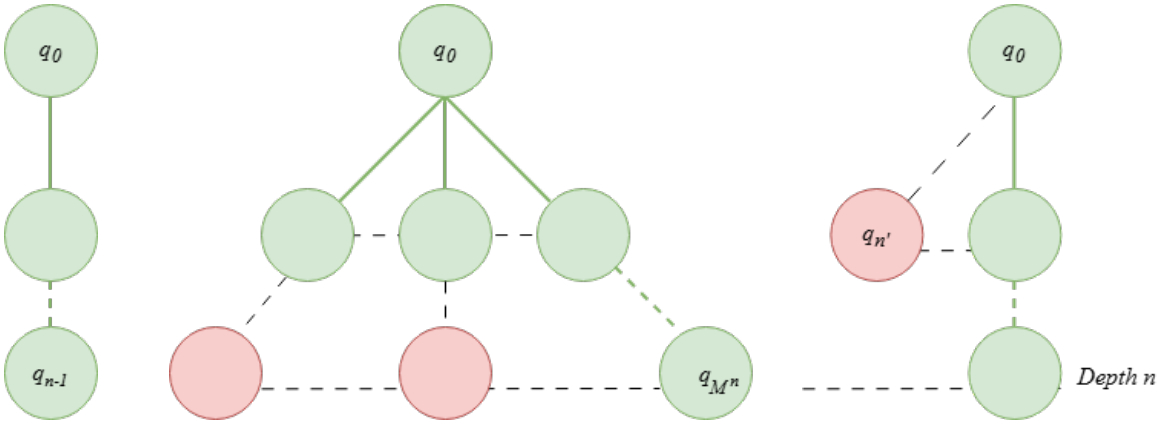
Algorithm 2 repeats until k candidate pitch sequences are generated.

The pitch generation algorithm generates k candidate pitch sequences, each with n pitches according to the node sequence generated by the note generation algorithm. Each pitch is selected according to the pitch transition model and structure label. The number of pitches in the pitch transition model is a constant that is independent of k and n. In the best case, only one branch of the tree is expanded, so the time complexity is $O(n)$. Thus, the best-case time complexity of the algorithm to generate k sequences is $O(kn)$.

In the worst case, all branches of the tree have to be explored before k sequences are accepted. For a tree of depth n whose nodes each having M child nodes, it has a total of M^n nodes. Therefore, the worst-case time complexity is $O(M^n)$. This worst-case scenario seldom happens because sequence rejections usually occur earlier, say no longer than depth $n' < n$. Thus, a typical time complexity is $O(M^{n'} + kn)$, which reduces to either $O(M^{n'})$ or $O(kn)$, depending on which term dominates. Fig. 4.1 illustrates best case, worst case and typical case scenarios of the pitch generation algorithm.

4.3.5 Melody Generation

The melody generation algorithm generates a melody of desired structure as specified by the user. The music structure includes the melody key, key signature,



(a) Best Case **(b)** Worst Case **(c)** Typical Case

Figure 4.1: Tree structures illustrating the best case, worst case, and typical case scenarios of the pitch generation algorithm.

number of bars m, bar length and structure labels. It applies the note and pitch generation algorithms to generate melody for each bar, one at a time. Recall that pitch generation algorithm generates k candidate pitch sequences for each bar by locally maximizing the pitch transition probabilities, subject to structure constraints. Among these k candidate sequences, the melody generation algorithm chooses the one with the largest confidence. The confidence is measured as sum of the transition probability of each pitch p_j in pitch class c_j to pitch p_{j+1} in pitch class c_{j+1} in the direction δ_j as defined in Section 4.2. Therefore, the confidence C is

$$C = \sum_{j=1}^{n} P(c_{j-1}, c_j, \delta_j). \tag{4.4}$$

The sequence with the largest confidence C is the one that best matches the style of the reference music.

Algorithm 3 Melody Generation Algorithm

Input: Target melody structure, pitch transition model.
Output: Target bars h_i, $i = 0, \ldots, m-1$

1: **for** $i = 0, \ldots, m-1$ **do**
2: **if** $\lambda_i = i'$ **then**
3: Copy the melody of $h_{i'}$ to h_i.
4: **else**
5: **if** $i = 0$ **then**
6: Randomly select an initial pitch p'.
7: **else**
8: Let p denote the last pitch of bar h_{i-1}.
9: Select p' with largest transition probability from p.
10: Apply note generation algorithm to generate note sequence.
11: Apply pitch generation algorithm to generate k pitch sequences.
12: Among the k sequences, select the one with the largest confidence C (Eq. 4.4).
13: Assign the selected pitch sequence to the note sequence to form the bar h_i.

The melody generation algorithm is straightforward. It generates one bar at a time as follows:

- Lines 2, 3: If bar h_i repeats bar $h_{i'}$ exactly, then the melody of $h_{i'}$, which includes the note sequence and pitch sequence, is copied to h_i.
- Lines 5 to 9: An appropriate initial pitch is selected for bar h_i. For the first bar of the melody, Line 6 generates an initial pitch randomly. Alternatively, it can also use an initial pitch provided by the user. For subsequent bars, the pitch with the largest transition probability from the last pitch of the previous bar is selected.
- Line 10 applies the note generation algorithm to generate a note sequence according to the note model and structure constraints.
- Line 11 applies the pitch generation algorithm to generate k candidate sequences according to the pitch transition model and structure constraints.
- Line 12 selects the best sequence among the k sequences.
- Line 13 assigns the selected pitch sequence to the generated note sequence to form bar h_i.

This algorithm is repeated for each bar of the target melody.

Algorithm 3 iterates m times for m bars. In each bar, it applies the note generation algorithm and the pitch generation algorithm to generate the note sequence and pitch sequences. The note generation algorithm has time complexity of $O(n)$. The pitch generation algorithm has best-case, worst-case and typical time complexity of $O(kn)$, $O(M^n)$ and $O(M^{n'} + kn)$, respectively, with $n' < n$. Therefore, the melody generation algorithm has best-case, worst-case and typical time complexity of $O(kmn)$, $O(mM^n)$ and $O(mM^{n'} + kmn)$, respectively.

Chapter 5

Experiments and Discussion

This chapter presents experiments that illustrate the performance of the melody generation algorithm. Audio files of the results are available in https://tinyurl.com/3mpwyje7. It is organized into two main parts, namely note generation tests and pitch generation tests. Note generation tests examine the note generation algorithm. For clear visualization of the results in the form of Western music scores, the generated note sequences are paired with known pitch sequences. Pitch generation tests examine the pitch generation algorithm. The generated results are also paired with known pitch sequences for visualization in the form of music scores.

Note that the melodies of songs can appear in a wide variety of forms. Whether a melody sounds pleasing depends on the listeners' preference. There is no objective metric for measuring the goodness of a generated melody. Therefore, quantitative comparisons are omitted; only qualitative comparisons with respect to the reference songs are made. Qualitative comparisons of the proposed method with two known DNN methods are illustrated Section in 5.3.

5.1 Note Generation Tests

5.1.1 Note Generation Test with Single Reference

This section illustrates the generation of note sequences with a single reference song. The target note styles include the pop style and jazz style, each derived from a single reference song. In particular, the pop-style reference song is *Over*

the Rainbow and the jazz-style reference song is *Fly Me to the Moon*.

1. Data Preparation

A pop-style and a jazz-style note model were derived from each of the two reference songs. The note probability $P(t,d)$ was computed by counting the frequency of occurrence of each (t,d) pair. In addition, the most frequently occurring note duration, denoted by d^*, and the latest onset time t_l at which a long note with duration $d > d^*$ occurs were also computed. Finally, ε was computed by counting the frequency of bars that begin with a long note of duration $d > d^*$.

For visualization in the form of music score, a pitch sequence was extracted from *Over the Rainbow*, a pop song, and *Fly Me to the Moon*, a jazz song. Two melodies were generated as follows to illustrate the flexibility of the algorithms:

1. Pop-style note sequence with pop-style pitch sequence.
   Pop-style note model was used along with the pitch sequence of *Over the Rainbow*.

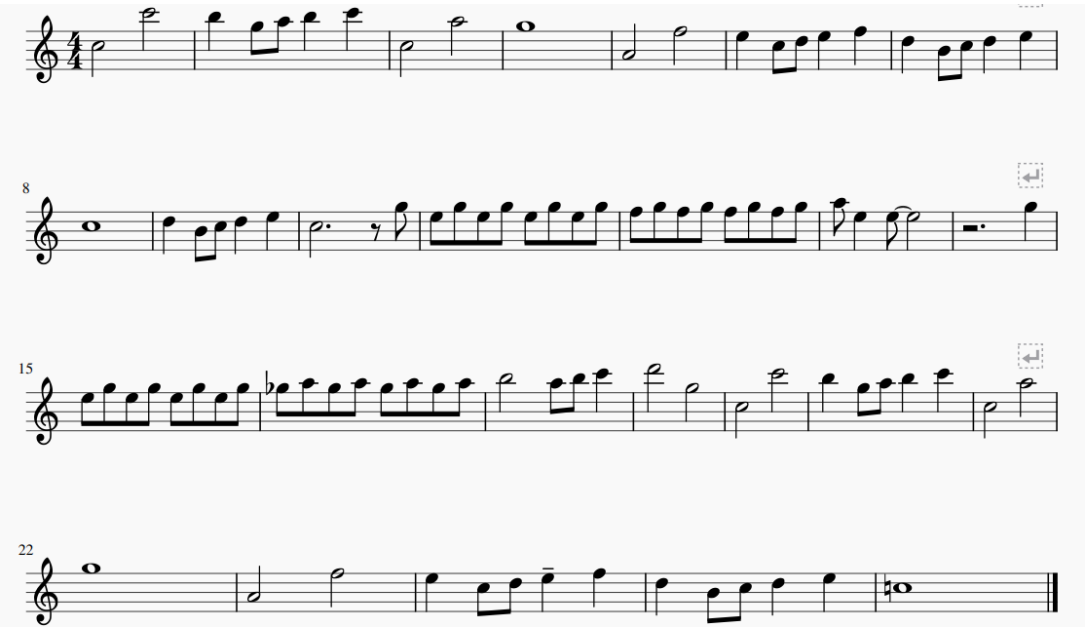
2. Jazz-style note sequence with jazz-style pitch sequence.
   Jazz-style note model was used along with the pitch sequence of *Fly Me to the Moon*.

For the note generation algorithm, each bar to be generated was assigned a structure label λ to reflect a simple and common musical form.

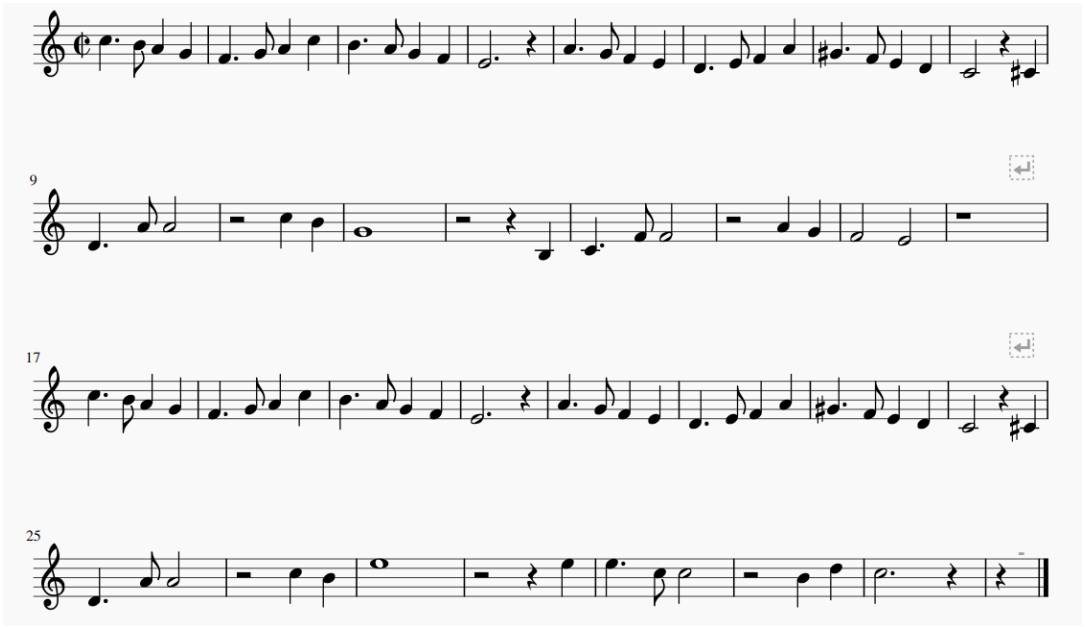
2. Test Results and Discussions

Figure 5.2 illustrates the first 8 bars of the generated melodies, with the corresponding structure label for each bar shown above the bar. The note sequence in Fig. 5.2(a) follows the characteristics common in pop style. It is regular and consists mostly of simple durations like quarter and half notes. On the other hand, the note sequence in Fig. 5.2(b) follows the characteristics common in jazz style. It has syncopation, dotted notes, and uneven patterns, which are typical in jazz style.

Furthermore, the styles of the reference songs are reflected in the generated results. Reference song *Over the Rainbow* consists of phrases with a long note at the beginning, at the end, or at both positions (Fig. 5.1(a)). Fig. 5.2(a) reflects these characteristics: bars 2, 5, and 7 start with long notes, bar 6 ends with a long note, and bars 1 and 3 both start and end with long notes. This shows that the algorithm is able to generate long notes at the beginning, end, or both positions of a bar when such notes exist in the note model. In addition, bars 4 and 8 satisfy

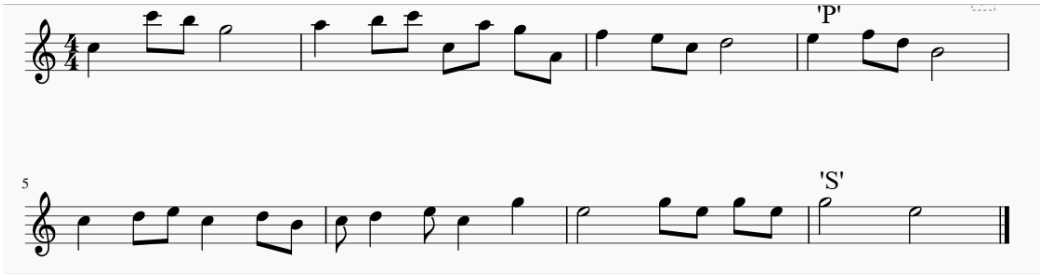


(a)

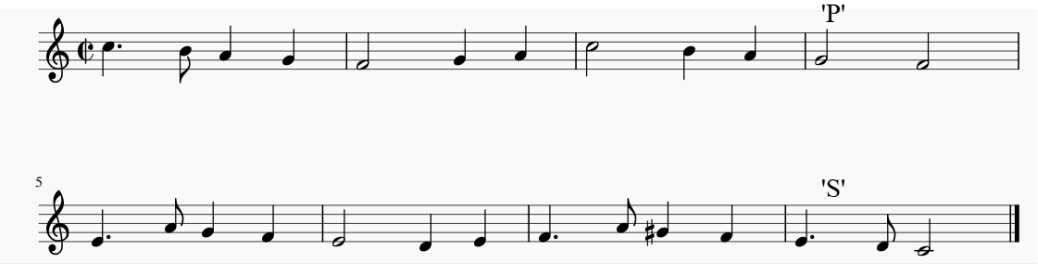


(b)

Figure 5.1: Reference songs used to derive note models. (a) Pop-style reference song, *Over the Rainbow*. (b) Jazz-style reference song, *Fly Me to the Moon*.



(a)



(b)

Figure 5.2: Note generation from single reference. (a) Pop-style pitch sequence with pop-style note sequence. (b) Jazz-style pitch sequence with jazz-style note sequence.

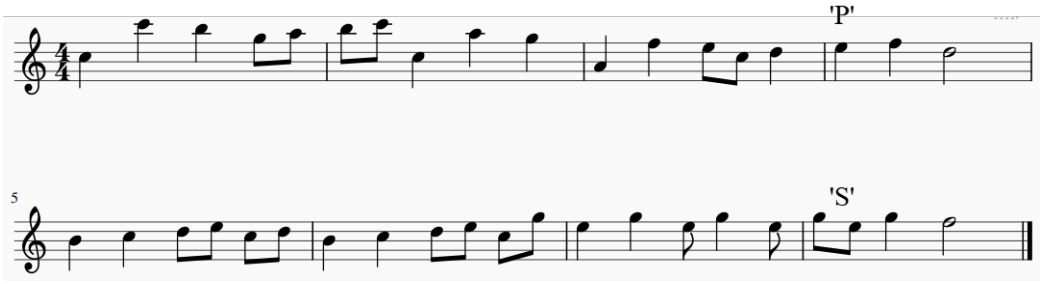
the closure condition, with the final long note placed on a strong beat. Reference song *Fly Me to the Moon* consists mostly of phrases that start with long notes (Fig. 5.1(b)). Fig. 5.2(b) reflects the same style, with most bars starting this way. Fig. 5.2(b) also shows that bars 4 and 8 satisfy the closure condition, with the final long note placed on a strong beat.

5.1.2 Note Generation Test with Multiple References

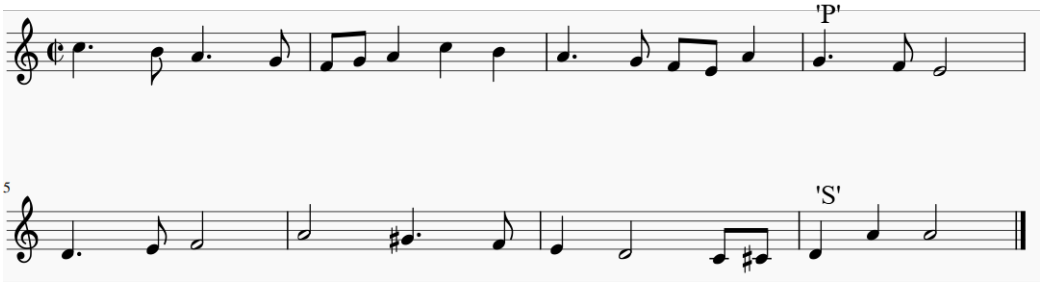
This section illustrates the generation of note sequences with multiple reference songs. The target pitch styles include the pop style and jazz style, each derived from a set of reference songs.

1. Data Preparation

Two sets of songs, one for pop and one for jazz, were extracted from Chord Melody Dataset (CMD) [60]. Each set contained 20 songs. A pop-style and a



(a)



(b)

Figure 5.3: Note generation from multiple references. (a) Pop-style pitch sequence with pop-style note sequence. (b) Jazz-style pitch sequence with jazz-style note sequence.

jazz-style note model was derived from each of the two sets following the same procedure described in Section 5.1.1.

As for Section 5.1.1, two pitch sequences were used, namely *Over the Rainbow*, a pop song, and *Fly Me to the Moon*, a jazz song.

2. Test Results and Discussions

Figure 5.3 illustrates the first 8 bars of the generated melodies with the corresponding structure label for each bar shown above the bar. Similar to Fig. 5.2(a), the note sequence in Fig. 5.3(a) is regular and consists mostly of simple durations. In contrast, similar to Fig. 5.2(b) the note sequence in Fig. 5.3(b) has syncopation, dotted notes, and uneven patterns, which are typical of jazz style.

5.2 Pitch Generation Tests

5.2.1 Pitch Generation Test with Single Reference

This section illustrates the generation of pitch sequences with a single reference song. The target pitch styles include the pop style and jazz style, each derived from a single reference song. In particular, the pop-style reference song is *Beauty and the Beast* and the jazz-style reference song is *Fly Me to the Moon*.

1. Data Preparation

A pop-style and a jazz-style pitch transition model was derived from each of the two reference songs. Pitch transition probability $P(c, c', \delta)$ was computed by counting the frequency of occurrence of pitch transition from class c to class c' in direction δ.

For visualization in the form of music score, a note sequence was extracted from *Beauty and the Beast*, a pop song, and *Fly Me to the Moon*, a jazz song. Three melodies were generated as follows to illustrate the flexibility of the algorithms:

1. Pop-style pitch sequence with pop-style note sequence.
   Pop-style pitch transition model was used along with the note sequence of *Beauty and the Beast*.

2. Jazz-style pitch sequence with pop-style note sequence.
   Jazz-style pitch transition model was used along with the note sequence of *Beauty and the Beast*.

3. Jazz-style pitch sequence with jazz-style note sequence.
   Jazz-style pitch transition model was used along with the note sequence of *Fly Me To The Moon*.

For the pitch generation algorithm, the number M of pitches with the largest transition probabilities was set to 3, and the number k of candidate pitch sequences generated was set to 5. Moreover, the acceptance parameters were set to $\alpha = \beta = 2$. Furthermore, for each bar to be generated, a structure label λ was assigned to reflect a simple and common musical form.

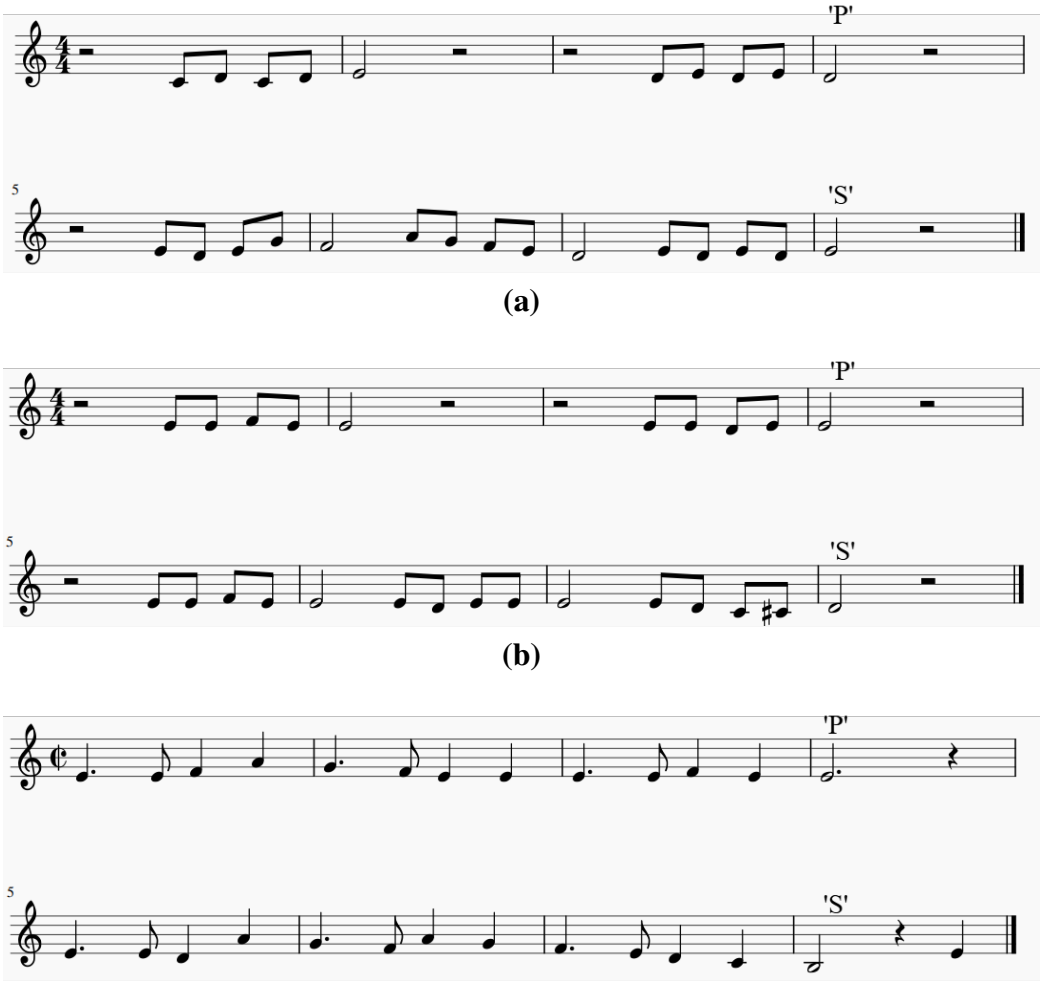


Figure 5.4: Pitch generation from single reference. (a) Pop-style pitch sequence with pop-style note sequence. (b) Jazz-style pitch sequence with pop-style note sequence. (c) Jazz-style pitch sequence with jazz-style note sequence.

2. Test Results and Discussions

Figure 5.4 illustrates the first 8 bars of the generated melodies with corresponding structure label for each bar shown above the bar. The melodies in Fig. 5.4(a) and 5.4(b) have the same note sequence because they are generated using the same note sequence of *Beauty and the Beast*. On the other hand, the melody in Fig. 5.4(c) has a different note sequence compared to those of Fig. 5.4(a) and 5.4(b) because it is generated using the note sequence of *Fly Me to the Moon*, which has syncopated rhythm.

The pitch sequence in Fig. 5.4(a) has the characteristics typical of pop-style, such as stepwise transitions between consecutive scale notes and repeated notes that give the melody a stable and straightforward quality. In contrast, the pitch sequences in Fig. 5.4(b) and in Fig. 5.4(c) have the characteristics common in jazz style, including one-semitone pitch transitions and notes outside the main scale that introduce tension. However, while melody in Fig. 5.4(b) has these pitch characteristics, its rhythm does not reflect jazz style, whereas melody in Fig. 5.4(c) maintains the jazz style more fully. This comparison shows that both pitch sequence and rhythm are necessary to maintain a musical style.

5.2.2 Pitch Generation Test with Multiple References

This section illustrates the generation of pitch sequences with multiple reference songs. The target pitch styles include the pop style and jazz style, each derived from a set of reference songs.

1. Data Preparation

Two sets of songs, one for pop and one for jazz, were extracted from Chord Melody Dataset (CMD) [60]. Each set contained 20 songs. A pop-style and a jazz-style pitch transition model was derived from each of the two sets following the same procedure described in Section 5.2.1.

As for Section 5.2.1, two note sequences were used, namely *Beauty and the Beast*, a pop song, and *Fly Me to the Moon*. The parameters of the algorithms are the same as those in Section 5.2.1.

2. Test Results and Discussions

Figure 5.5 illustrates the first 8 bars of the generated melodies with corresponding structure label for each bar shown above the bar. As for Fig. 5.4, melodies Fig. 5.5(a) and 5.5(b) have the same note sequence because they are generated using the same note sequence. In contrast, melody Fig. 5.5(c) has a different jazz-style syncopated rhythm.

Similar to Fig. 5.4(a), Fig. 5.5(a) stays within the scale, with no semitone transitions or notes outside the scale. In contrast, similar to Fig. 5.4(b) and Fig. 5.4(c), the pitch sequences in Fig. 5.5(b) and Fig. 5.5(c) include semitone transitions or notes outside the scale. Moreover, Fig 5.5(c) has jazz-style rhythm along with its pitch sequence, so it maintains the jazz style better than does Fig 5.5(b).

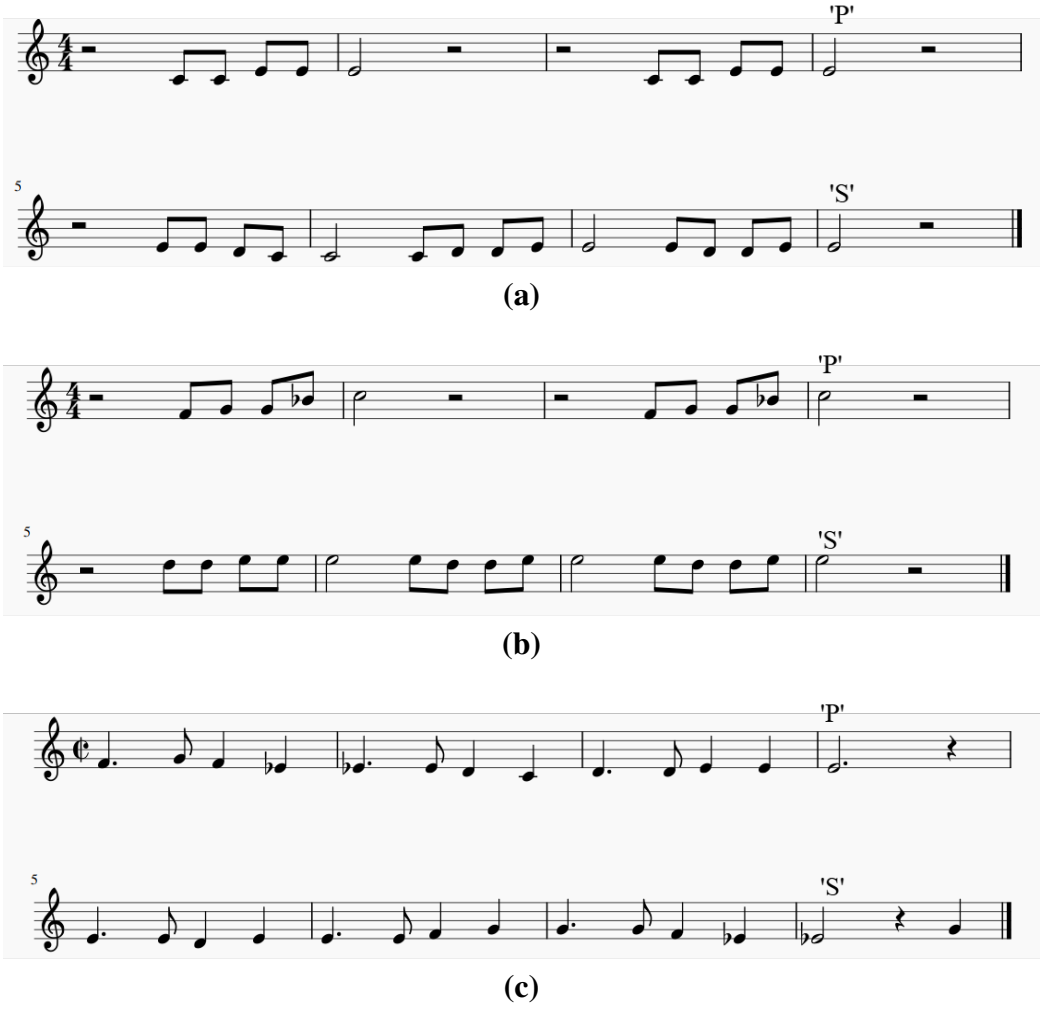
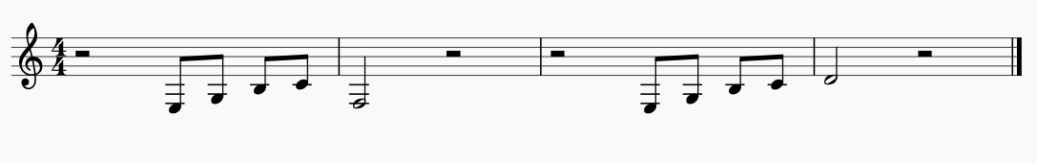


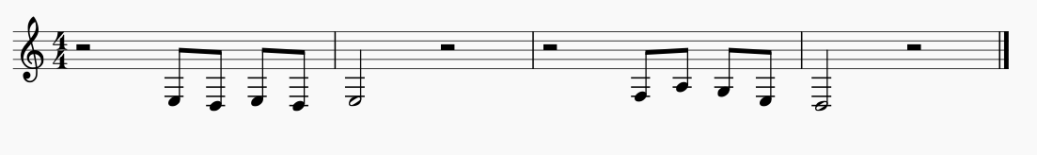
Figure 5.5: Pitch generation from multiple references. (a) Pop-style pitch sequence with pop-style note sequence. (b) Jazz-style pitch sequence with pop-style note sequence. (c) Jazz-style pitch sequence with jazz-style note sequence.

5.2.3 Exhaustive Generation of Pitch Sequence

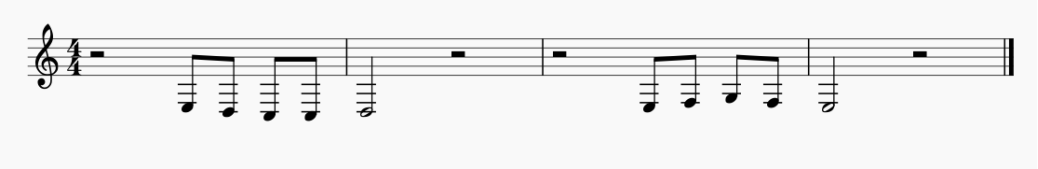
The pitch generation algorithm is designed to generate k candidate pitch sequences for the melody generation algorithm to select. In the extreme, it can generate all possible candidate pitch sequences exhaustively. To illustrate this property, a test was conducted using *Beauty and the Beast* for the note sequence as well as the pitch transition model. To reduce execution time, only the first 4 bars of the melodies were generated.



(a)



(b)



(c)

Figure 5.6: Examples from exhaustive generation of pitch sequences. (a) Pitch sequence that matches the reference song *Beauty and the Beast* exactly. (b), (c) Variations of pitch sequences.

The pitch generation algorithm generated a total of 11,623 4-bar pitch sequences. One of these generated pitch sequences matches the reference song *Beauty and the Beast* (Fig. 5.6(a)). Two other generated pitch sequences are shown in Fig. 5.6(b) and 5.6(c). These pitch sequences are very similar to each other, and can be considered as variations of each other. These results show that the algorithm can generate all possible pitch sequences, including the reference sequence, and other variations derived from the pitch transition model.

5.3 Full-length Melody Generation

5.3.1 Full-length Melody Generation with Single Reference

This section illustrates full-length melody generation with a single reference song, with emphasis on section contrast and closure. The reference song is *I Want to*

Hold Your Hand, which consists of two contrasting sections, namely section A and section B.

1. Data Preparation

Two note models were derived, one from each section of the reference song. In addition, two pitch transition models were derived, one from each section of the reference song.

Two sections of a melody were generated as follows:

- 8-bar section in the style of section A of the reference song.
- 8-bar section in the style of section B of the reference song.

In both cases, style refers to rhythmic style and pitch style.

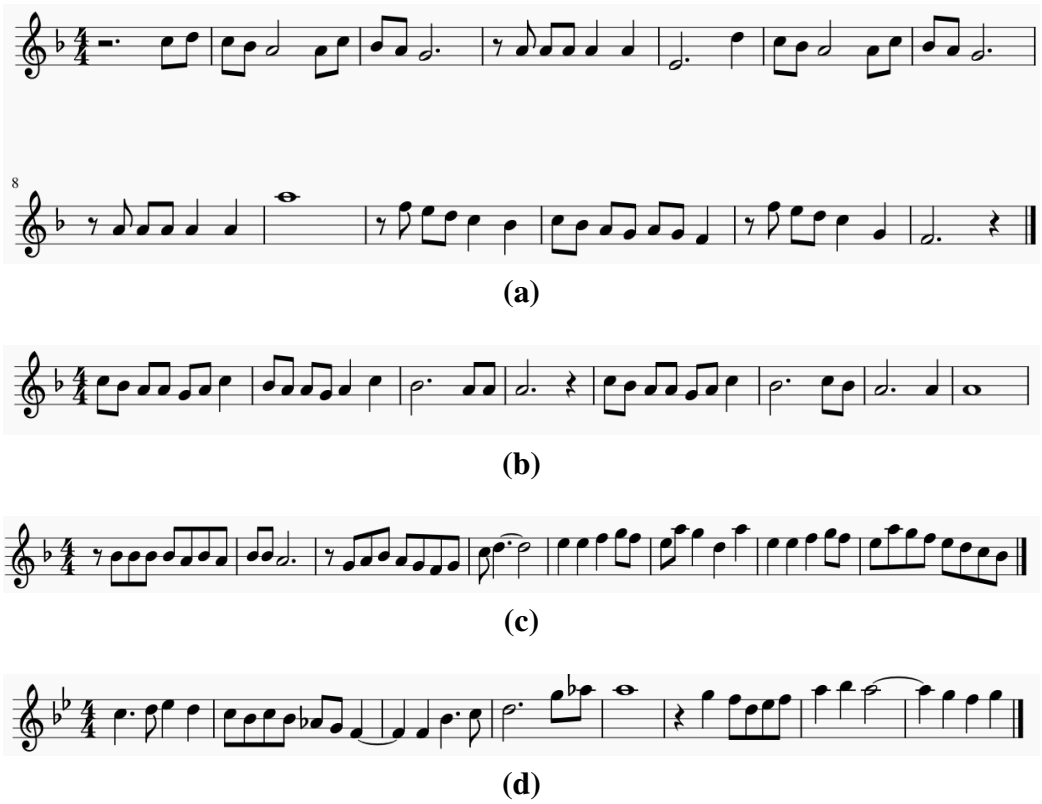
For comparison, two DNN models of Google Magenta [61] were used, namely Magenta LookbackRNN model and Magenta AttentionRNN model. Both models take a prompt melody as input and generate a continuation of the specified length, providing a basis for comparison in melody generation.

A section in the style of section A was generated by providing section A of the reference song as the prompt melody to the DNN model. Similarly, a section was generated in the style of section B by using section B of the reference song as the prompt melody. The same procedure was applied to both DNN models, so each model produced two sections.

2. Test Results and Discussions

The reference song *I Want to Hold Your Hand* consists of two sections. Section A (Fig. 5.7(a)) has a dynamic rhythm with high note density, frequent changes in note durations, and a dynamic melodic contour with ascending and descending pitch transitions. On the other hand, section B (Fig. 5.8(a)) has a stable rhythm and a stable melodic contour, creating a contrast with the variability of section A.

Fig. 5.7(b), (c), and (d) illustrate the results generated by the proposed method, Magenta LookbackRNN, and Magenta AttentionRNN, respectively. The melody in Fig. 5.7(b) has a dynamic rhythm and contour that closely follow the style of the reference section A, with well-defined phrasing and clear closures at bars 4 and 8. In contrast, Fig. 5.7(c) shows a rhythmic style that differs from the reference, with a more dynamic rhythm. Fig. 5.7(d) is stylistically closer to the reference than does Fig. 5.7(c), yet Fig. 5.7(b) remains the most similar to the reference overall.



(a)

(b)

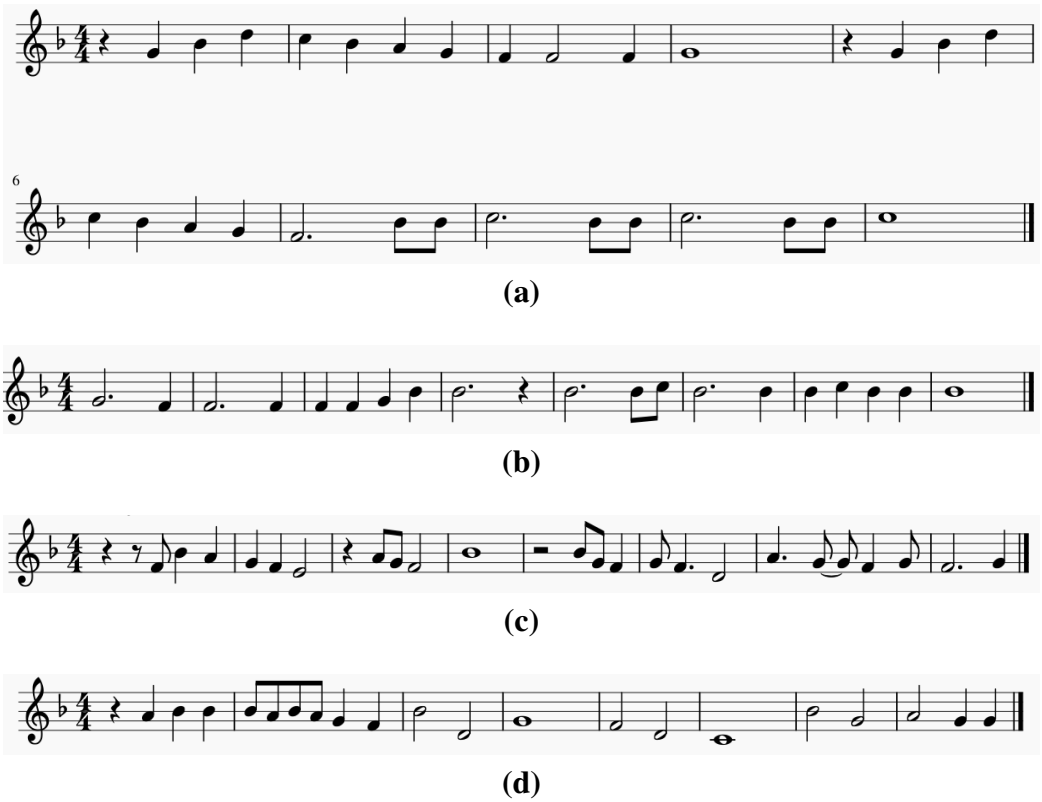
(c)

(d)

Figure 5.7: Section A of the reference song, *I Want to Hold Your Hand*, and sections in the style of section A of the reference song, generated by different methods: (a) Section A of the reference song. (b) Result of the proposed method. (c) Result of Magenta LookbackRNN. (d) Result of Magenta AttentionRNN.

Furthermore, both Fig. 5.7(c) and Fig. 5.7(d) lack clear phrasing and closure at the end of the section.

Fig. 5.8(b), (c), and (d) illustrate the results generated by the proposed method, Magenta LookbackRNN, and Magenta AttentionRNN, respectively. The melody in Fig. 5.8(b) has a stable rhythm and a melodic contour that follows the style of the reference section B, with well-defined phrasing and clear closures at bars 4 and 8. On the other hand, Fig. 5.8(c) has a different style than the reference, with a more dynamic rhythm and a more dynamic contour than the reference. Fig. 5.8(d) also has a different rhythmic and melodic style than the reference, particularly toward the end of the section. As noted in the discussion of generation results in section A style, both Fig. 5.8(c) and Fig. 5.8(d) lack clear phrasing and closure at the section ending.



(a)

(b)

(c)

(d)

Figure 5.8: Section B of the reference song, *I Want to Hold Your Hand*, and sections in the style of Section B of the reference song, generated by different methods: (a) Section B of the reference song. (b) Result of the proposed method. (c) Result of Magenta LookbackRNN. (d) Result of Magenta AttentionRNN.

5.3.2 Full-length Melody Generation with Two References

This section illustrates full-length melody generation using two reference songs: *I Want to Hold Your Hand* as the rhythmic style reference and *Love Me Tender* as the pitch style reference. Both reference songs consist of two contrasting sections, namely section A and section B.

1. Data Preparation

Two note models were derived, one from each section of the rhythmic style reference. Furthermore, two pitch transition models were derived, one from each section of the pitch style reference song.

Two sections of a melody were generated as follows:

- 8-bar section:
  Note sequence in the style of section A of rhythmic style reference.
  Pitch sequence in the style of section A of pitch style reference.

- 8-bar section:
  Note sequence in the style of section B of rhythmic style reference.
  Pitch sequence in the style of section B of pitch style reference.

2. Test Results and Discussions

The pitch style reference song *Love Me Tender* consists of two sections. Section A (Fig. 5.9(a)) contains no repeated pitches, resulting in a dynamic melodic contour with ascending and descending transitions. In contrast, section B (Fig. 5.9(c)) includes repeated pitches, which lead to a stable melodic contour.

Fig. 5.9(b) and Fig. 5.9(d) illustrate the sections generated by the proposed method in the style of section A and section B, respectively. The rhythm in Fig. 5.9(b) is the same as in Fig. 5.7(b), since both were generated using the same rhythmic style reference. Similarly, the rhythm in Fig. 5.9(d) is the same as in Fig. 5.8(b). In terms of pitch, Fig. 5.9(b) has a dynamic contour with no repeated pitches, consistent with the style of section A of the pitch style reference song. By contrast, Fig. 5.9(d) has a stable contour with repeated pitches, reflecting the style of section B of the pitch style reference song.

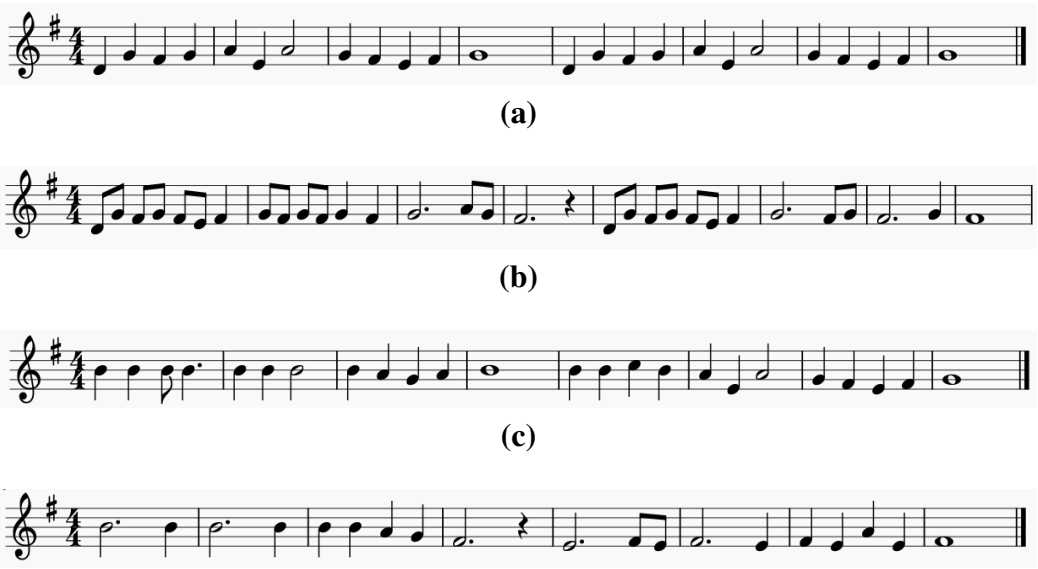


Figure 5.9: Pitch style reference song, *Love Me Tender*, and sections generated by the proposed method: (a) Section A of the pitch style reference song. (b) Result of the proposed method generated according to reference section A. (c) Section B of the pitch style reference song. (d) Result of the proposed method generated according to reference section B.

Chapter 6

Conclusion

This thesis presented a novel method for style-specific melody generation, inspired by music artist's creative process of referencing existing music to guide stylistic decisions. The proposed method consists of three parts, namely note generation, pitch generation and melody generation. The note generation algorithm generates a sequence of notes within a bar. The pitch generation algorithm generates k candidate pitch sequences for a bar. The melody generation algorithm selects one of the k pitch sequence to generate a melody of the desired structure, as specified by the user, by applying the note and pitch generation algorithms to each bar.

The proposed method enables a user to generate melodies that reflect the stylistic characteristics of as few as one reference piece. The core contributions of this work are threefold. First, it introduces a melody generation method that captures stylistic features of rhythm and pitch from one or more reference pieces and supports style-specific generation of note sequence and pitch sequence. Second, it can adapt to various music styles without retraining or manual rule definition. Third, it employs training-free direct optimization, eliminating the need for large datasets and simplifying the generation process. Therefore, it is an efficient and flexible method for generating melodies of various user-specified styles.

Test results show that the algorithms can generate note sequences and pitch sequences of various styles, producing interesting melodies. In particular, the pitch generation algorithm can generate all possible pitch sequences, including the original sequence of the reference song from which the pitch transition model is derived. It can also generate variations of the melodies that are similar in profile but differ in some pitch details. Test results also show that the results generated

by the proposed method follow the reference styles better than those generated by two known DNN methods.

Despite its strengths, the proposed methods have some shortcomings.

- The current algorithms work on one bar of music at a time. For many reference songs, phrasing occurs over two bars instead of one bar. The current algorithms can be easily extended to handle these styles by changing "bar" to "fragment", which consists of 2 bars.
- The current note model captures probabilities $P(t,d)$ over onset and duration but omits information from preceding notes. Extending the note model to include such dependencies could improve note sequence coherence.
- User control is limited in the current algorithms. A possible improvement is to allow the user to specify rhythmic density, melodic contour and fluctuation to better guide the algorithms.
- The melody generation algorithm supports repetition and phrase or section closure and basic mechanisms for variation and contrast. Enhancing control over these aspects within and across sections would allow the generated melodies to have clearer structural patterns and better organization of musical form.

References

[1] K. Ebcioğlu. “An expert system for harmonizing four-part chorales”. In: *Computer Music Journal* 12.3 (1988), pp. 43–51.

[2] É. Gilbert and D. Conklin. “A probabilistic context-free grammar for melodic reduction”. In: *Proceedings of the International Workshop on Artificial Intelligence and Music, 20th International Joint Conference on Artificial Intelligence*. 2007, pp. 83–94.

[3] D. Cope. *Computers and musical style*. Vol. 6. Oxford University Press Oxford, 1991.

[4] D. Cope. *The algorithmic composer*. Vol. 16. AR Editions, Inc., 2000.

[5] C. Roig et al. “Automatic melody composition based on a probabilistic model of music style and harmonic rules”. In: *Knowledge-Based Systems* 71 (2014), pp. 419–434.

[6] T. Tanaka, B. Bemman, and D. Meredith. “Constraint programming approach to the problem of generating milton babbitt’s all-partition arrays”. In: *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings 22*. Springer. 2016, pp. 802–810.

[7] G. Aguilera et al. “Automated generation of contrapuntal musical compositions using probabilistic logic in derive”. In: *Mathematics and Computers in Simulation* 80.6 (2010), pp. 1200–1211.

[8] D. Quick and P. Hudak. “Grammar-based automated music composition in Haskell”. In: *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design*. 2013, pp. 59–70.

[9] D. Quick and P. Hudak. “A temporal generative graph grammar for harmonic and metrical structure”. In: *ICMC*. 2013.

[10] D. Quick. “Learning production probabilities for musical grammars”. In: *Journal of New Music Research* 45.4 (2016), pp. 295–313.

[11] R. F. Cádiz. “Creating music with fuzzy logic”. In: *Frontiers in Artificial Intelligence* 3 (2020), p. 59.

[12] I. Paz-Ortiz et al. “Using the Fuzzy Inductive Reasoning methodology to improve coherence in algorithmic musical beat patterns”. In: *Artificial Intelligence Research and Development*. IOS Press, 2014, pp. 289–292.

[13] R. R. Spangler. *Rule-based analysis and generation of music*. California Institute of Technology, 1999.

[14] G. D. Smith et al. “Genetic algorithm utilising neural network fitness evaluation for musical composition”. In: *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Norwich, UK, 1997*. Springer. 1998, pp. 219–223.

[15] J. Jeong, Y. Kim, and C. W. Ahn. “A multi-objective evolutionary approach to automatic melody generation”. In: *Expert Systems with Applications* 90 (2017), pp. 50–61.

[16] Brian Martínez-Rodríguez. “A new fitness function for evolutionary music composition”. In: *International Conference on Mathematics and Computation in Music*. Springer. 2022, pp. 205–217.

[17] E. Özcan and T. Erçal. “A genetic algorithm for generating improvised music”. In: *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer. 2007, pp. 266–277.

[18] G. Papadopoulos, G. Wiggins, et al. “A genetic algorithm for the generation of jazz melodies”. In: *Proceedings of STEP* 98 (1998).

[19] Y. Wen and C. Ting. “Composing bossa nova by evolutionary computation”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[20] F. Mo et al. “A music generation model for robotic composers”. In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2018, pp. 1483–1488.

[21] R. De Prisco, G. Zaccagnino, and R. Zaccagnino. “EvoComposer: an evolutionary algorithm for 4-voice music compositions”. In: *Evolutionary computation* 28.3 (2020), pp. 489–530.

[22] C. De Felice et al. “Splicing music composition”. In: *Information Sciences* 385 (2017), pp. 196–212.

[23] Bill Manaris et al. “A corpus-based hybrid approach to music analysis and composition”. In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 22. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 2007, p. 839.

[24] M. D. Johnson, D. R. Tauritz, and R. W. Wilkerson. “Evolutionary computation applied to melody generation”. In: *Intelligent Engineering Systems Through Artificial Neural Networks* (2004).

[25] M. Marques et al. “Music composition using genetic evolutionary algorithms”. In: *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*. Vol. 1. IEEE. 2000, pp. 714–719.

[26] B. Stoltz and A. Aravind. “MU_PSYC: music psychology enriched genetic algorithm”. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2019, pp. 2121–2128.

[27] Z. Zeng and L. Zhou. “A memetic algorithm for Chinese traditional music composition”. In: *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*. IEEE. 2021, pp. 187–192.

[28] Fernando H Calderon Alvarado et al. “Melody Similarity and Tempo Diversity as Evolutionary Factors for Music Variations by Genetic Algorithms.” In: *ICCC*. 2020, pp. 251–254.

[29] Andrew Gartland-Jones. “Can a genetic algorithm think like a composer”. In: *Generative Art*. 2002.

[30] L. R. de Azevedo Santos, C. N. Silla Jr, and MD Costa-Abreu. “A methodology for procedural piano music composition with mood templates using genetic algorithms.(2021)”. In: (2021).

[31] R. Loughran and M. O’Neill. “Clustering agents for the evolution of autonomous musical fitness”. In: *Computational Intelligence in Music, Sound, Art and Design: 6th International Conference, EvoMUSART 2017, Amsterdam, The Netherlands, April 19–21, 2017, Proceedings 6*. Springer. 2017, pp. 160–175.

[32] T. Collins et al. “Developing and evaluating computational models of musical style”. In: *AI EDAM* 30.1 (2016), pp. 16–43.

[33] T. Collins and R. Laney. “Computer-generated stylistic compositions with long-term repetitive and phrasal structure”. In: *Journal of Creative Music Systems* 1.2 (2017).

[34] M. Farbood and B. Schöner. “Analysis and synthesis of Palestrina-style counterpoint using Markov chains”. In: *ICMC*. 2001.

[35] I. Goienetxea et al. “Statistics-based music generation approach considering both rhythm and melody coherence”. In: *IEEE Access* 7 (2019), pp. 183365–183382.

[36] D. Herremans et al. “Generating structured music for bagana using quality metrics based on Markov models”. In: *Expert Systems with Applications* 42.21 (2015), pp. 7424–7435.

[37] C.F. Huang et al. “Analyzing the perception of Chinese melodic imagery and its application to automated composition”. In: *Multimedia Tools and Applications* 75 (2016), pp. 7631–7654.

[38] F. Pachet. “The continuator: Musical interaction with style”. In: *Journal of New Music Research* 32.3 (2003), pp. 333–341.

[39] F. Pachet et al. “Finite-length Markov processes with constraints”. In: *transition* 6.1/3 (2001).

[40] F. Pachet and P. Roy. “Markov constraints: steerable generation of Markov sequences”. In: *Constraints* 16.2 (2011), pp. 148–172.

[41] A. Papadopoulos, P. Roy, and F. Pachet. “Avoiding plagiarism in Markov sequence generation”. In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 28. 1. 2014.

[42] A. S. Ramanto and N. U. Maulidevi. “Markov chain based procedural music generator with user chosen mood compatibility”. In: *International Journal of Asia Digital Art and Design* 21.1 (2017), pp. 19–24.

[43] W. Schulze and B. Van Der Merwe. “Music generation with Markov models”. In: *IEEE MultiMedia* 18.03 (2011), pp. 78–85.

[44] T. Kathiresan. “Automatic Melody Generation”. Master’s thesis. Stockholm, Sweden: KTH Royal Institute of Technology, 2015. URL: `https://kth.diva-portal.org/smash/record.jsf?pid=diva2:864281`.

[45] M. McVicar, S. Fukayama, and M. Goto. “AutoLeadGuitar: Automatic generation of guitar solo phrases in the tablature space”. In: *2014 12th international conference on signal processing (ICSP)*. IEEE. 2014, pp. 599–604.

[46] D. Williams et al. “AI and automatic music generation for mindfulness”. In: *2019 AES International Conference on Immersive and Interactive Audio: Creating the Next Dimension of Sound Experience*. York. 2019.

[47] F. Colombo, A. Seeholzer, and W. Gerstner. “Deep artificial composer: A creative neural network model for automated melody generation”. In: *Computational Intelligence in Music, Sound, Art and Design: 6th International Conference, EvoMUSART 2017, Amsterdam, The Netherlands, April 19–21, 2017, Proceedings 6*. Springer. 2017, pp. 81–96.

[48] G. Hadjeres, F. Pachet, and F. Nielsen. “DeepBach: a Steerable Model for Bach Chorales Generation”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1362–1371.

[49] C. Z. A. Huang et al. “Counterpoint by Convolution”. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*. 2017.

[50] N. Jaques et al. “Tuning Recurrent Neural Networks with Reinforcement Learning”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[51] D. D. Johnson, R. M. Keller, and N. Weintraut. “Learning to Create Jazz Melodies Using a Product of Experts”. In: *Proceedings of the Eighth International Conference on Computational Creativity (ICCC)*. 2017, pp. 151–158.

[52] S. H. Hakimi, N. Bhonker, and R. El-Yaniv. “BebopNet: Deep Neural Models for Personalized Jazz Improvisations.” In: *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*. 2020, pp. 828–836.

[53] C. Payne. *MuseNet*. `https://openai.com/index/musenet/`. Accessed: 2025-07-08. 2019.

[54] L. Kawai, P. Esling, and T. Harada. “Attributes-Aware Deep Music Transformation”. In: *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*. 2020.

[55] A. Roberts et al. “A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. PMLR, 2018, pp. 4364–4373.

[56] R. Yang et al. “Deep Music Analogy via Latent Representation Disentanglement”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*. 2019, pp. 596–603.

[57] L. Yu et al. “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient”. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*. 2017, pp. 2852–2858.

[58] Peiling Lu et al. “MeloForm: Generating Melody with Musical Form based on Expert Systems and Neural Networks”. In: *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*. 2022.

[59] S. Dai et al. “Controllable Deep Melody Generation via Hierarchical Music Structure Representation”. In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 143–150.

[60] S. Hiehn. *Chord melody dataset*. `https://github.com/shiehn/chord-melody-dataset`.

[61] E.Waite et al. *Project Magenta: Generating Long-Term Structure in Songs and Stories*. `https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn/`. 2016.