Towards Human-Like Music Intelligence via Concept Alignment

by

Ziyu Wang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science
New York University
August, 2025

Gus G. Xia

Dedication

To the joyful and curious seekers—

in music, in science, and in life

Acknowledgements

For a young newcomer stepping into the field of computer music, everything began with curiosity. At the time, it was impossible to imagine what this journey would come to mean—or how colorful and transformative it would become. Now, as I complete this dissertation, I realize how truly meaningful this period has been. It offered me a space—comfortable and unburdened—to grow again and better understand myself: my body, my energy, my thoughts, and my way of expression. This process of self-reflection is far from complete, but it has progressed just enough for me to continue discovering what I am passionate about and what I hope to contribute.

It is for this reason that I want to express my gratitude to everyone I've met and everything I've encountered, all of which, in their own way, brought me here. I especially thank my parents, Min Yu and Jian Wang, for bringing me into this world and gently guiding me, with patience and trust, along a steady and thoughtful path. I also thank Maoran Xu, who was once my undergraduate classmate and is now a professor in statistics, for sharing her creative soul, encouraging me to pursue research, and introducing me to my future Ph.D. advisor. Finally, I thank all my teachers, classmates, and friends from the schools I attended and the orchestras and music clubs I was part of. This is how the journey began.

I would like to express my deepest gratitude to my Ph.D. advisor, Prof. Gus Xia, whose support has been invaluable at the most critical points of my doctoral journey. It has been a true joy to discuss research with Gus—he embodies the spirit that research life, with all its challenges, thrives on emotion and passion. As the captain of Music X Lab, Gus also gave me the opportunity to work with a wonderful and collaborative crew. Over time, I found myself not only growing within this environment but also gradually taking on more responsibility, shaping research directions, and supporting others along the way. I'd like to thank Yiyi Zhang, Ruihan Yang, Junyan Jiang, Ke Chen, Tianyao Chen, Kai Shao, Yixiao Zhang, Dingsu (Derek) Wang, Jingwei Zhao, Daniel Chin, Liwei Lin, Shiqi Wei, Yichen Huang, Leyi (Vera) Sun, Ningran Song, Yutian Qin, Yang Qu, Qinying Lei, Ziyue Piao, Xuanjie Liu, Kun Fang, Lejun Min, Qi He, Yuxuan Wu, Xinyue Li, Wenye Ma, Ahmad Hammoudeh, Muhammad Taimoor Haseeb, Chenxi Bao, Xiaoyu Yi, Longshen Ou, and Haobo Yang. We made the lab a truly joyous place to be. There is nothing more fulfilling than working together, only for the work and nothing else.

Beyond Music X Lab, I have been fortunate to receive mentorship and support from Prof. Roger B. Dannenberg and Prof. Yann LeCun. Both served on my committee, witnessed my growth over the years, and generously offered insights from their respective perspectives in music and AI. I am also deeply grateful to the rest of my committee: Prof. Eric Xing, for hosting me at MBZUAI and offering his bold and visionary advice; Prof. He He and Prof. Cheng-Zhi Anna Huang, for their consistent encouragement and support throughout the final stages of my Ph.D. I feel lucky to have received support from many others, including Prof. Shlomo Dubnov, Dr. Akira Maezawa, Prof. Ichiro Fujinaga, Prof. Zhiyao Duan, Prof. Alexander Lerch, Prof. Zeyu Zheng, Prof. Julius Smith, Prof. Junbo (Jake) Zhao, Prof. Chris Donahue, Prof. Hao-Wen Dong, Prof. Keith Ross, Prof. Nasir Memon, Prof. Kun Zhang, Prof. Xiaosong Ma, Prof. Elizabeth Churchill, Prof. Mingming Gong, Prof. Samuel Horvath, Ms. Dekyi (Hong) Liang, Prof. Dasaem Jeong, Prof. Juhan Nam, Dr. Ying Shan, Dr. Wei-Hsiang Liao, and Mr. Yukara Ikemiya. I am also thankful to colleagues and peers such as Ke Chen, Yusong Wu, Shuqi Dai, Ruibin Yuan, Yinghao Ma, Lele Liu, Huan Zhang, Yujia Yan, Xiaoxuan Wang, Meiying Ding, Tingyu Zhu, and Haoyu Liu for their help and companionship throughout this journey.

During my studies at NYU Shanghai, NYU, and MBZUAI, I have been fortunate to meet many valuable people. I would like to thank Prof. Meiling Chen and Prof. Murray James Morrison from NYU Shanghai for their music lessons and guidance—both as teachers and as friends. I am also grateful for the support and friendship of fellow students: Yiming Zhang, Watcher Wang, Sean Welleck, Autumn (Yanqiu) Wu, Zhaochen Li, Yanling Deng, and Zhi Li from NYU Shanghai; Siqi Wang, Leyi Zhu, Jinkun Lin, Tao Wang, and Xiangyu Gao from NYU; and Yiqing Li, Longkang Li, Shuxian Zou, Gongxu Luo, Fathinah Asma Izzati, Nicolas Mauricio Cuadrado Avila, Jiaqi Ruan, and Jia He from MBZUAI. I would also like to thank the staff who helped ensure a smooth academic journey: Vivien Du, Eric Mao, Dr. Congyi Lv, Ran Mei, and Maggie Mao from NYU Shanghai; Santiago Pizzini from NYU; and Maria Pereira and Analye Mata from MBZUAI.

Lastly, I offer my heartfelt thanks to my girlfriend, Kun Fang. With her calm presence, graceful warmth, and quiet yet resolute strength, she has entered my life carrying a bright spirit, inspiring me to embrace perspectives I might never have discovered on my own. She has revealed to me the colors of life, with its joy, its richness, and its style, in the most intuitive and genuine way. I am deeply grateful for her support, care, encouragement, and patience throughout the most critical and uncertain moments of this journey.

I would like to conclude this acknowledgment by extending my gratitude once more to all that I have experienced and to everyone I have come to know.

Abstract

Recent advances in artificial intelligence (AI) have enabled music generation systems that can produce fluent and stylistically convincing pieces within seconds. Yet, despite its plausibility at the surface level, such systems often feel musically hollow—unable to explain their choices, adapt to nuanced prompts, or engage in meaningful creative exchange with humans. From a scientific perspective, this limitation arises because current models, guided primarily by data-driven objectives, lack the ability to understand and manipulate the underlying concepts necessary for music-making.

This dissertation introduces the framework of *concept alignment*, the methodologies that enable machines to manipulate musical concepts in ways that resemble human creative processes. Rather than treating deep generative models as “black boxes”, concept alignment reveals and structures the internal mechanisms of these models in terms of human-understandable concepts, enabling more interpretable and controllable AI systems for music creation.

The work focuses on three core capacities. First, *concept representation* investigates whether models can internalize music concepts without explicit labels, through disentanglement methods that learn representations such as pitch contour, accompaniment texture, and audio groove. Second, *concept organization* addresses how concepts are structured and coordinated across a composition. I develop hierarchical models that depict the generation process both in terms of its temporal elaboration (compositional hierarchy) and its progression from abstract to concrete music ideas (concept hierarchy). Third, *concept emergence* explores whether new concepts can arise naturally from data. I propose a domain-general framework that utilizes statistical inductive biases to guide models in discovering symbolic structures, such as pitch classes or visual digits, directly from raw inputs like audio or images.

Across these three directions, the dissertation develops and evaluates methods that enable deep generative models to be aligned with human concept manipulation. The results demonstrate that with appropriate architectures and learning objectives, models can exhibit behaviors consistent with human-like concept manipulation, achieving controllability, interpretability, and generalizability in downstream applications. This offers both practical benefits for music creation and deeper insights into the computational foundations of creativity.

CONTENTS

Dedication **iii**

Acknowledgments **iv**

Abstract **vii**

List of Figures **xiii**

List of Tables **xx**

List of Appendices **xxiii**

1 Introduction **1**

1.1 Concept Manipulation in Music-Making 2

1.2 Background: How AI Handles Concepts 5

1.3 Concept Alignment 7

1.4 Dissertation Outline 9

I Concept Representation **11**

2 Disentangling Pitch and Rhythm in Monophonic Music **12**

2.1 Analogy-Making and Disentanglement 14

2.2 Methodology 16

2.3 Experiments 19

2.4 Related Work 27

2.5 Summary 29

3 Disentangling Chord and Texture in Polyphonic Accompaniment 30

3.1 Disentanglement for Controllable Music Generation 30

3.2 Model 32

3.3 Controlled Music Generation 36

3.4 Experiments 39

3.5 Related Work 43

3.6 Summary 44

4 Multimodal Disentanglement for Audio-to-Symbolic Music Generation 46

4.1 Audio-to-Symbolic Arrangement 47

4.2 Method 49

4.3 Experiments 52

4.4 Related Work 55

4.5 Summary 56

5 Disentangling Style in Large Language Models 57

5.1 Representation Learning via Bootstrapping 58

5.2 Method 60

5.3 Experiments 64

5.4 Related Work 72

5.5 Summary 73

II Concept Organization 74

6 Compositional Hierarchy 75

6.1 Compositional Hierarchy in Music Composition 76

6.2 Methodology 78

6.3 Analysis of Structural Music Generation 84

6.4 Experiments 84

6.5 Related Work 89

6.6 Summary 91

7 Concept Hierarchy 92

7.1 The TOMI Framework 92

7.2 Methodology 94

7.3 Experiment 98

7.4 Related Work 105

7.5 Summary and Future Work 106

III Concept Emergence 108

8 Concept Emergence via Unsupervised Disentanglement 109

8.1 Emergence and Content-Style Disentanglement 110

8.2 Methodology 113

8.3 Experiments 117

8.4 Related Work 124

8.5 Limitation 125

8.6 Summary 125

9 Conclusion **127**

9.1 Implicit Alignment of Multimodal Dynamics 128

9.2 Explicit Alignment in Cross-Domain Emergence 129

9.3 Human-AI Co-Creation 130

Appendices **131**

Bibliography **200**

List of Figures

1.1 Concepts and concept manipulation in music composition. Solid rectangles represent explicit concepts; dashed rectangles indicate implicit concepts that are not explicitly defined; and cloud shapes denote unnameable and deeply implicit concepts. Colored blocks indicate concept groups. 4

2.1 A comparison between vanilla sequence VAE [272] and our model with condition and disentanglement. 15

2.2 A comparison between Δz_{p} and Δz_{r} after transposition. 20

2.3 Evaluating the disentanglement by data augmentation. 21

2.4 The source melody. 22

2.5 The first example of analogy via replacing z_{p}. 22

2.6 The second analogy example via replacing z_{p}. 23

2.7 The first example of analogy via replacing z_{r}. 23

2.8 The second analogy example via replacing z_{r}. 24

2.9 Two examples of replacing the original chord. 24

2.10 An illustration of two-way interpolation. 25

2.11 Subjective evaluation results. 27

3.1 The model diagram of polyphonic disentanglement. 33

3.2 An example of compositional style transfer of 16-bar-long samples. 37

3.3 Examples of texture variations via posterior sampling and prior sampling. 38
3.4 An example of accompaniment arrangement conditioned on melody, chord progression, and first 2 bars of accompaniment. 39
3.5 Results of objective measurement. 41
3.6 Subjective evaluation results. Here "TFRed: xth largest" denotes the x^{th} (largest) order statistic of the transferred segments. 42

4.1 The proposed model architecture and training strategy. 49
4.2 An example of automatic arrangement based on the audio of an 8-bar excerpt from *1001 Nights* by Samuel Tai. 53
4.3 Subjective evaluation results. 55

5.1 A Q-Former module bridges the modality gap between a frozen audio LLM and a symbolic music LLM. It extracts cross-modal music *style* from the hidden representations of the audio LLM and, together with a lead sheet providing music *content*, conditions the symbolic LLM for piano arrangement. The Q-Former is trained in a two-stage process, effectively bootstrapping audio-to-symbolic arrangement without re-training either LLM backbone. 59
5.2 The Q-Former is a Transformer encoder with two parallel, modality-specific streams that share the self-attention layers. The left stream queries music style from an audio LLM via cross-attention, and the right stream encodes symbolic music. Cross-modal interactions are regulated by self-attention masks specific to training objectives. 62
5.3 A symbolic music LLM is used to generate piano arrangement based on two inputs: the query output $\mathbf{Z}$ from the Q-Former with cross-modal music style, and a lead sheet providing music content. A LoRA adapter is incorporated to reweight self-attention and accommodate the newly introduced lead sheet condition. . . . 64

5.4 Audio-to-symbolic arrangement for an 8-bar excerpt from *The Sound of Music*. Figures 5.4(b) to 5.4(d) are arranged based on the lead sheet in 5.4(a) and an audio reference from the original soundtrack, a ragtime piece, and a bossa nova piece, respectively. Preserved music contents are highlighted in blue note heads. Synthesized audio is submitted in supplementary. 67
5.5 Subjective evaluation on music quality, conducted on Ballroom and GTZAN datasets to assess the musicality for diverse music genres and styles. 70

6.1 The diagram of cascaded diffusion models for hierarchical symbolic music generation. 81
6.2 An example of whole-song generation of 40 measures under a given Form (A♭ major key and `"i4A4A4B8b4A4B8o4"` phrases). The three staves (from top to bottom) show the generated Reduced Lead Sheet, Lead Sheet, and Accompaniment. Here, rectangles with colored background are used to indicate the appearance of the same motifs in verse and chorus sections. Dashed border rectangles with colored background indicate a variation of motifs. We use red dotted rectangles to show where the generated score shows a strong implication of phrase boundary or cadence. The generated chord progressions in Reduced Lead Sheet and Lead Sheet are identical, shown by the chord symbols. 85
6.3 Subjective evaluation results on music quality and well-structuredness. GT indicates ground truth samples composed by humans. 88

7.1 Concept hierarchy in TOMI: music ideas developed from features to clips are transformed and integrated into the composition, organized by sections and tracks. 93

7.2 We show the structured LLM output in (a) with the corresponding music generation process in (b). We distinguish node types by color and shape, with a detailed attribute example for each type shown on the left of (a). We depict three composition link examples as colored arrows in (b), highlighting their results with rectangles in corresponding colors. 95

7.3 ILS similarity matrices example, where all four compositions are generated under the section sequence: *i* (intro), *v1* (verse 1), *p1* (pre-chorus 1), *c1* (chorus 1), *v2* (verse 2), *p2* (pre-chorus 2), *c2* (chorus 2), *b* (bridge), *c3* (chorus 3), and *o* (outro). Darker colors indicate higher segment similarity. The blocks marked as yellow-edge boxes are same-label similarities, with diagonal values marked as red lines being excluded, and the remaining parts are different-label similarities. 101

7.4 Subjective evaluation results of mean score with within-subject confidence intervals, where *p1-p4* corresponds to four parts of our survey: *p1. Local Music Quality*, *p2. Consistency Among Same-Label Phrases*, *p3. Contrast Between Different-Label Phrases*, and *p4. Overall Full-Song Evaluation*. 104

8.1 An illustration of the variance-versus-invariance constraints of content and style. 111

8.2 The model architecture of V3. Left: The autoencoder has two branches for content and style, respectively, where the content branch has a VQ layer at the encoder output. Right: the V3 constraints, where double-dashed arrows represent measuring the variability by $\nu_k(\cdot)$, and solid arrows represent taking the average. 114

8.3 Comparison of generated images by recombining $\boldsymbol{z}^{\mathrm{s}}$ from given sources in SVHN and all $\boldsymbol{z}^{\mathrm{c}}$ in the learned codebook. 123

A.1 An illustration of the proposed polyphonic syntax. 133

A.2 An illustration of PianoTree data structure to encode the music example in Figure A.1. 137

A.3 An overview of the model architecture. The recurrent layers are represented by rectangles and the fully-connected (FC) layers are represented by trapezoids. The `note`, `simu_note`, and `score` events are represented by circles. 139

A.4 A visualization of `note` embeddings after dimensionality reduction using PCA. . . 144

A.5 A visualization of `simu_note` embeddings after dimensionality reduction using PCA. 145

A.6 Subjective evaluation results of latent space interpolation. 146

A.7 Subjective evaluation results of downstream music generation. 147

A.8 Long music generations given first 4 measures. 148

B.1 The diagram of the proposed melody reduction algorithm. 152

B.2 An illustration of post-processing operations. 159

B.3 Subjective evaluation results of melody reduction quality across three genres. . . 161

B.4 Comparison of the original melody, melody reductions from the proposed method, and the baselines. We highlight the phrases in the top row. 161

B.5 Comparison of variations of an example melody. Here **Diff.** denotes the conditional diffusion model trained to generate melody variations from melody reductions. Positive comments are highlighted in red, negatives in blue. 163

B.6 Subjective results of melody variations. 163

C.1 An example data representation of our proposed hierarchical music language. . . 166

C.2 Examples of generated Reduced Lead Sheet of `"A8"` phrase in E♭ major. The samples marked with * are controlled by the external condition, which represents "unchanging chord progression." 170

C.3 Examples of generated Lead Sheet of `"A8"` phrase in E♭ major given the upper-level Reduced Lead Sheet Figure C.3a. The samples marked with * are controlled by the external condition, which represents "sixteenth-note rhythm." 171

C.4 Examples of generated Accompaniment of `"A8"` phrase in E♭ major given the upper-level Lead Sheet Figure C.4a. The sample marked with * is controlled by the external condition, which represents “Alberti bass.” 172

C.5 The music samples for external control. 173

D.1 Left: example training data in PhoneNums. Right: example data for out-of-distribution evaluation in PhoneNums. 181

D.2 Example data in the original SVHN dataset. The digits in the images are bounded by the red boxes. 184

D.3 t-SNE visualization of the learned digit (content) and color (style) representations on PhoneNums when there is no codebook redundancy ($K = 10$). 188

D.4 t-SNE visualization of the learned digit (content) and color (style) representations on PhoneNums when the codebooks are redundant. 188

D.5 t-SNE visualization of the learned pitch (content) and timbre (style) representations on InsNotes when there is no codebook redundancy ($K = 12$). 189

D.6 t-SNE visualization of the learned pitch (content) and timbre (style) representations on InsNotes when the codebooks are redundant. 189

D.7 t-SNE visualization of the learned content (digit) representations on SVHN. 190

D.8 t-SNE visualization of the learned content (action) representations on Sprites. The 10 colors refer to 10 different actions. 190

D.9 Confusion matrices of learned codebooks on PhoneNums. The horizontal axes show digit labels from “0” to “9”, and the vertical axes show codebook atoms sorted by ground truth digit labels. 191

D.10 Confusion matrices of learned codebooks on InsNotes. The horizontal axes show pitch labels from “C” to “B”, and the vertical axes show codebook atoms sorted by ground truth pitch labels. 192

D.11 Confusion matrices of learned codebooks on SVHN. The horizontal axes show digit labels from “0” to “9”, and the vertical axes show codebook atoms sorted by ground truth digit labels. 193
D.12 Confusion matrices of learned codebooks on Sprites. The horizontal axes are different action labels, and the vertical axes show codebook atoms sorted by ground truth action labels. 193
D.13 Confusion matrices of learned codebooks on Libri100. The horizontal axes are different phoneme labels, and the vertical axes show codebook atoms sorted by ground truth phoneme labels. 194
D.14 Comparison of generated digits by recombining content and style latents using unsupervised methods trained on PhoneNums. 195
D.15 An illustration of the correspondence between the content-style duality and the Yin-Yang duality. 198

List of Tables

2.1 The evaluation results of pitch- and rhythm-wise augmentation-based query. . . . 22

5.1 Objective evaluation on audio-to-symbolic coherence. Ballroom and GTZAN serve as out-of-distribution datasets to assess cross-domain arrangement capabilities. 69

5.2 Ablation study on the impact of individual pre-training objectives to cross-modal representation learning. **C**, **M**, and **G** denote the use of the Audio-to-Symbolic Contrastive, Matching, and Generative losses, respectively. 71

6.1 Definition of the four-level hierarchical music language. We use `m` for measure, `b` for beat, `s` for step, to represent the temporal resolution. M denotes the number of measures in a piece, γ denote the number of beats in a measure, and δ denotes the number of steps in a beat. 79

6.2 Objective evaluation of music structure via the proposed inter-phrase latent similarity. 87

7.1 Statistics of sample databases. 100

7.2 Objective evaluation results of FAD with two models and ILS with three latent representations. 101

8.1 Evaluation of digit and color disentanglement on PhoneNums using latent retrieval. Values are reported in percentage. 119
8.2 Evaluation of pitch and timbre disentanglement on InsNotes using latent retrieval. Values are reported in percentage. 120
8.3 Linear probing accuracies (in %) for content (digit) classification on SVHN. 121
8.4 Linear probing accuracies (in %) for content (action) classification on Sprites. . . . 121
8.5 Content classification accuracies (in %) on data with OOD styles. 121
8.6 Quantitative results of codebook interpretability on datasets with discrete style labels. Values are reported in percentage. 122
8.7 Quantitative results of codebook interpretability on datasets without style labels. Values are reported in percentage. 123

A.1 Objective evaluation results on reconstruction criteria. PianoTree is our proposed method. Other columns correspond to the baseline models described in Section A.4.2. 143

C.1 Definition of phrase type 167
C.2 Configuration of the conditioning methods in four stages of the proposed model. 168
C.3 The hyperparameter configuration of diffusion model training. The listed attributes are the same across all four stages. 168
C.4 Objective evaluation of external control efficacy of chord, rhythm, and texture in the three diffusion stages. $\mathrm{dis}^{\mathrm{rb}}$ denotes the rule-based distance-based metric and $\mathrm{sim}^{\mathrm{lt}}$ denotes the latent similarity-based metric. 174
C.5 Evaluation on whether the generative models copy the training data. The highlighted data in red indicates potential copying the training set. 176

C.6 Theoretical comparison of time and model parameter complexity between the cascaded approach and the end-to-end approach. η denotes the resolution scaling factor. L denotes the time scope (i.e., receptive field) of cascaded models. 179

D.1 List of colors and their corresponding RGB values. Colors only used for out-of-distribution experiments in Section 8.3.3 are marked with *. 181

D.2 List of instruments and their corresponding MIDI program numbers. Instruments only used for out-of-distribution generalization are marked with *. 182

D.3 List of phonemes with their indices. 185

D.4 Ablation study of V3 settings on content-style disentanglement performance on PhoneNums. Values are reported in percentage. 196

D.5 Ablation study of V3 settings on content-style disentanglement performance on InsNotes. Values are reported in percentage. 196

List of Appendices

A PianoTree VAE for Polyphonic Representation Learning 132
B Automatic Melody Reduction 149
C Supplementary Materials for Chapter 6 165
D Supplementary Materials for Chapter 8 180

1 Introduction

We live in an age where artificial intelligence (AI) refines our essays, plans our schedules, plays games with us, and even drives our cars. Music, too, has joined the trend. AI systems can now compose pieces in seconds. Yet, unlike other domains where these technologies translate into intuitive applications, the experience of using AI in music composition often leaves us unfulfilled.

Consider a few familiar scenarios:

- You give the model a few measures from the opening of a Bach prelude and ask it to continue. It produces several fluent versions—each different, yet all somehow shallow and arbitrary. Which one should you choose? Do any of them actually mean anything?

- You prompt the model to compose a piece in a happy mood. You sample many times, only to realize that the model treats "happy" as nothing more than "fast tempo and major key," which is far short of the emotional depth you had in mind.

- You offer a creative idea like "make a techno-pipa track," a prompt that blends timbre and genre. The model responds with techno sounds and something vaguely resembling pipa, but the nuance is gone. After repeated attempts, you abandon the exploration.

In all these cases, music AI feels hollow, as if every output were equally plausible, equally indifferent. The model never defends its choices, never explains what it was trying to express, nor how your input has shaped its direction. Where is the *musicianship*? And how can such an interaction ever fulfill our creative or emotional needs?

Maybe we are asking too much of AI. After all, music is part of what makes us human. But that's precisely why music AI is such a challenging problem of intelligence. Like other AI systems, music models are trained to match patterns in data. Yet the actual aim of music-making goes far beyond pattern imitation. As an art form, music requires exploration, expression, and interaction—music AI should be able to propose ideas, to respond to subtle prompts, to become creative partners who can discuss music with us in meaningful ways. These goals are inherently abstract and hard to quantify. However, if AI scientists must define something close to measurable, our goals should not just be about musical output, but about the *underlying process*—whether AI can understand and act on the underlying meaning of music in a human-like way.

This thesis explores one path toward that goal by treating musical concepts as the fundamental elements underlying the process of music-making. These concepts range from concrete features such as pitch and timbre, to more nuanced ones like texture and style, and even to higher-level abstractions that shape the overall structure of a piece. I investigate whether music models can learn to manipulate such internal concepts in ways similar to humans, e.g., learning representations of concepts, organizing them into hierarchies, and discovering entirely new ones. I refer to this capability as *concept alignment*. Grounding music generation in concept alignment makes intention, interaction, and control over the concepts intrinsic to the model—qualities absent in systems focused solely on matching output patterns. While this work does not claim to fully answer the question of machine musicianship, it opens a way toward meeting human creative and emotional needs.

1.1 Concept Manipulation in Music-Making

Let us now see how concepts function in the process of music composition. Consider an over-simplified workflow: when composing a piece, one might begin with a rough idea and gradually elaborate it into finer details, drawing on concepts across different levels of granularity—much like building a house from a draft to finer details. This process is, of course, idealized, which implies levels of concepts from top to bottom. In reality, a piece may emerge in many ways: from a motif or chord progression at the start, or from some intuitive spark whose relation to existing theories seems unknown—purely emotional and personal. Yet in all these processes, levels of concepts are still present. Concepts are not a predefined dictionary but a fluid and evolving set. A motif or chord progression may first invoke low-level intervals, pitches, keys, and other concepts; soon, we may turn to high-level structural concerns, and then back again. Likewise, new "sparks" often become new concepts in themselves, which we carry forward and weave into the rest of the composition. Concepts can be numerous and nuanced, but their existence is undeniable, and they form a structure—not of compositional order, but of relationships and hierarchies between ideas—that persists throughout creation.

I have found it helpful to consider how these concepts can be broadly categorized. In particular, I distinguish between two types of concepts involved in music creation:

1. **Concepts *of* music**: structural and time-based elements within the piece itself (e.g., chord, pitch, phrase, form).

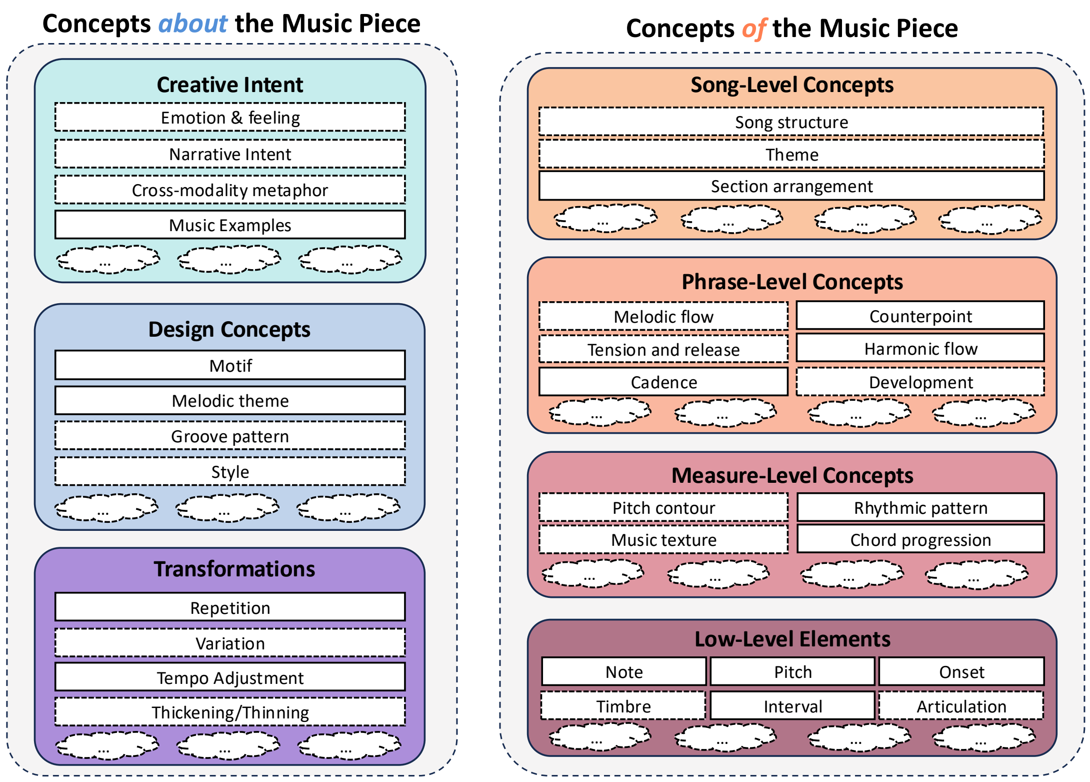
2. **Concepts *about* music**: external abstractions we use to describe or interpret music (e.g., melancholic, jazzy, minimalist, or Debussy-like).

From this distinction emerge two corresponding hierarchies:

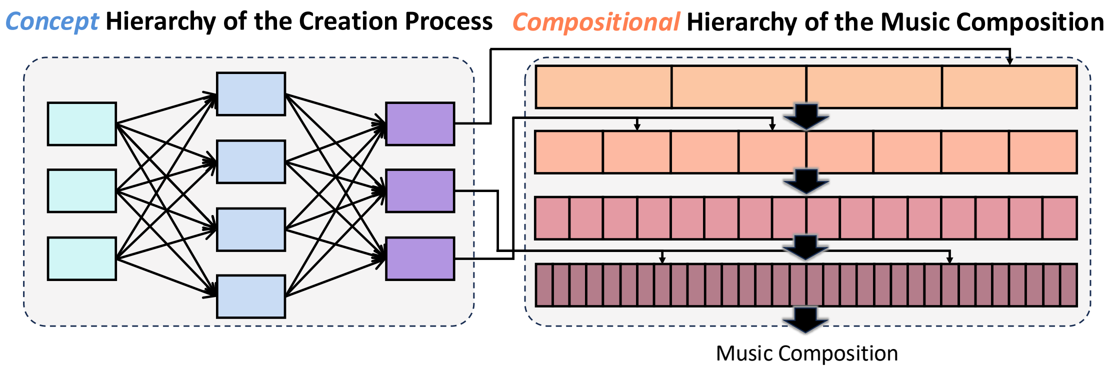
1. **Compositional hierarchy**: built from concepts of music. It organizes ideas across scales: notes form phrases, phrases form sections, and so on.

2. **Concept hierarchy**: built from concepts about music, connecting more abstract notions, such as emotion, style, or influence, to more concrete ones, such as themes and motifs, and showing how musical ideas are elaborated during composition.

Figure 1.1 offers a rough sketch of this layered space of concepts. In any finished composition, the compositional hierarchy shows how the music unfolds over time, while the concept



(a) Ontological categorization of music concepts.


Concept Hierarchy of the Creation Process
Compositional Hierarchy of the Music Composition
Music Composition

(b) Realization of concepts in hierarchical music generation.

Figure 1.1: Concepts and concept manipulation in music composition. Solid rectangles represent explicit concepts; dashed rectangles indicate implicit concepts that are not explicitly defined; and cloud shapes denote unnameable and deeply implicit concepts. Colored blocks indicate concept groups.

hierarchy guides and influences it. The compositional hierarchy is internal to the piece and uses concepts of music to carry on the progression of the piece at various time scales, whereas the concept hierarchy forms a knowledge graph of concepts about music and can be informed by prior works, textual prompts, or even other modalities, which is not detailed in the current figure. For human composers, these two hierarchies interact naturally. In the most simplified case, composition might begin by shaping the concept hierarchy—from an overall mood down to specific musical elements—which then grounds itself in music structure, giving rise to the compositional hierarchy. In more realistic cases, attention flows fluidly between the two hierarchies, whether consciously or not. In either case, composition involves manipulating concepts—encoding, organizing, comparing, combining, and discovering them. These concept manipulation capabilities are what we aim for AI to learn in a human-like way.

The diagram is intentionally rough: it is not meant to define precise terms or rigid categories, but to suggest how concepts might be organized in a human mind. Some concepts are explicit, such as cadence and swing rhythm, while others are implicit, such as pitch contour or texture—hard to name yet deeply felt. Many emerge only when triggered by examples or contexts, and some may never be verbalized at all. The hierarchy itself is also approximate, as the boundaries between levels are not fixed. Later in this thesis, I will refer to this diagram in more detail and highlight the aspects of concept manipulation most critical for machines to learn.

1.2 Background: How AI Handles Concepts

Having considered how concepts operate in human music creation, we now turn to how they are embedded in existing AI systems. Historically, prior to the deep learning era, AI research placed greater emphasis on white-box approaches, which explicitly encode known concepts and workflows into model design. Rule-based systems and small-scale, interpretable statistical models were designed to describe the composition process in explicit, human-readable terms [67, 213, 256]. However, such methods often fell short in capturing the full complexity and variability of musical patterns. Deep learning took a different path: rather than specifying concepts directly, it adopted a uniform architecture, the neural network, that learns to match patterns in data, with the expectation that relevant concepts would emerge implicitly. For example, in machine translation, deep learning models map inputs to outputs without relying on explicit linguistic rules [241]. Similarly, the music transformer trained on piano music may implicitly learn patterns of repetition and long-term dependency, such as attending more strongly to earlier similar passages, without being told what a "motif" is [106].

Although neural networks are composed of units that mimic neurons in our brains, this similarity remains at a low level. Human-like understanding of concepts is far from guaranteed. Even when deep generative models are trained on massive datasets and produce impressively realistic outputs, their grasp of concepts is often unstable. For example, they frequently struggle with geometry and physics, logical reasoning, self-consistency, and structures made of nested concept structures, such as the relationship between a human body and its parts, or between an orchestra and its sections and musicians [21, 255, 277].

Beyond their unstable grasp of concepts, current models also lack effective mechanisms for concept manipulation. A representative example is *analogy-making*—or, in music composition, *style transfer*—which requires identifying the appropriate level of abstraction, altering selected concepts, and keeping the rest intact and compatible. While supervised training is possible when paired examples exist, achieving this naturally and without supervision demands flexible concept manipulation capabilities. This thesis will demonstrate how enabling such capabilities can make style transfer possible, yet the overall problem remains a major challenge.

There are several relevant approaches for improving the transparency of neural networks, such as introducing inductive biases [13, 78, 125], applying regularization [11, 98, 218], incorporating causal and probabilistic frameworks [194, 212], and building world models that represent hierarchies and dynamics in the environment [84, 140, 266]. Similar to this thesis, these methods also aim to enhance interpretability, and some are adopted in this thesis. However, the focus here is distinct: the goal is to make AI systems behave more like humans, whereas these methods are general tools that can serve many other purposes.

1.3 Concept Alignment

The core idea of this thesis is to treat concept manipulation as the central mechanism driving the composition process. This differs from standard *controllable music generation*, where all concepts are predefined and models are simply conditioned on them. Here, I focus on concepts that are nuanced, hard to define, hierarchical, or emergent. I refer to the approach as concept alignment—the design of AI methodologies that enable artificial systems to manipulate concepts in ways that mirror human creative processes.

In this thesis, I examine three types of concept alignment problems: *representation*, *organization*, and *emergence*. Each corresponds to a key operation in human concept manipulation and involves challenges that make it nontrivial. These challenges are best understood in the context of the overall concept structure shown in Figure 1.1. The three problems are outlined below:

1. **Concept Representation**: Some musical concepts, like emotion, key, chord, or textual tags, can be labeled and embedded in neural networks as vectors. But many important concepts are implicit, such as "pitch contour," which we intuitively recognize but cannot easily define or annotate. These unlabeled, felt concepts are abundant in music. Concept learning addresses how to represent such concepts without relying on explicit labels, often requiring disentangling certain concepts from the whole.

2. **Concept Organization**: Music concepts are difficult to enumerate, yet, as illustrated in Figure 1.1, they naturally form compositional and concept hierarchies in the creative process. This thesis explores how such hierarchies can be defined and used in generative models without explicitly defining all underlying concepts. We demonstrate that capturing these hierarchical structures is essential for maintaining the structure of long-term music generation.

3. **Concept Emergence**: Not all concepts are predefined—some arise spontaneously from the

data or emerge only temporarily within specific contexts. This aspect of concept manipulation involves discovering new concepts under general learning assumptions, without relying on predefined musical or domain-specific knowledge. The key focus here is to develop domain-general inductive biases, enabling machines to form abstract representations from raw data alone.

Together, these three aspects form the core capabilities of concept manipulation. Each captures a distinct facet of how humans engage with ideas during the creative process and requires different modeling strategies and assumptions. Together, they enable controllability, interpretability, and the ability to create with intent, such as analogy-making, within the scope of the concepts studied in this work. This leads to the central claim of the thesis:

Thesis Statement. *This dissertation proposes that artificial intelligence can be designed to approximate human-like concept manipulation, particularly in terms of learning, organization, and emergence, so that it demonstrates controllability and interpretability in collaborative music creation with humans.*

Admittedly, concept manipulation in the human mind is far more complex than what this dissertation addresses. Even within the studies presented here, certain concepts, such as music texture, prove difficult to generalize across genres, and the emergence of more abstract or deeply nested concepts remains out of reach. It is also unclear how well these methods scale to larger or more complex models, and whether structural clarity must be traded off against generation quality. Finally, this approach inevitably simplifies many nuances of creative behavior: people manipulate concepts differently, experience modalities differently, and develop conceptual fluency across age, culture, and personal history. These aspects are not explicitly modeled in this thesis. Nonetheless, by framing creativity through the lens of concept alignment, this dissertation offers a concrete starting point for connecting AI generation with human creativity.

1.4 Dissertation Outline

This dissertation is organized into three parts, corresponding to three types of concept alignment and further divided into nine chapters in total, each addressing a specific aspect of the concept alignment problem. Below is an outline of the content and contributions of each chapter:

Chapter 1 outlines the concept alignment problem of this thesis and the background of music composition and artificial intelligence research.

Part I includes Chapters 2–5 and focuses on concept representation. Chapter 2 investigates concept representation in monophonic music. It introduces the EC^2-VAE model, which disentangles pitch contour and rhythm for interpretable control and style transfer. Chapter 3 extends the concept representation framework to polyphonic textures and chord progression. It presents a model for learning disentangled latent chord and texture representations of piano accompaniments. Chapter 4 considers representation disentanglement in a multimodal setting where a representation of audio texture is learned to control music generation. Chapter 5 further considers representation learning problems in a large language model (LLM) setting and uses cross-modal fine-tuning techniques to achieve learning piano accompaniment style from audio.

Part II includes Chapters 6–7 and focuses on concept organization. Chapter 6 addresses compositional hierarchy through the task of whole-song generation. It proposes a four-level hierarchical music language and a cascaded diffusion framework for structurally coherent composition. Specific concepts can be added at each level for additional control. Chapter 7 explores concept hierarchy in the context of electronic music generation. It defines a structured representation of a musical piece, illustrating how ideas can be selected, transformed, and arranged at specific positions within the composition. A text-based LLM is used to guide the generation of this structure with in-context learning.

Part III contains a single Chapter 8 and focuses on concept emergence. Chapter 8 investigates a fundamental problem in concept emergence: emerging content like “digits” or “pitches” from raw observations. It introduces a domain-general statistical inductive bias based on the variance and invariance of content and style. This bias enables the unsupervised disentanglement of latent factors and is applicable across modalities, for example, extracting digit and color in handwritten images or pitch and timbre in monophonic instrument audio.

Chapter 9 concludes the dissertation by summarizing key findings, open questions, and directions for future research on concept alignment.

Additional supporting materials, including extended experiments, implementation details, and supplementary figures, are provided in the appendices.

Part I

Concept Representation

2 | Disentangling Pitch and Rhythm in Monophonic Music

We begin our discussion of concept alignment with its most fundamental aspect: *concept representation*. For a given concept, representation refers to the ability of a model to form an internal structure that corresponds to that concept and influences the model's behavior accordingly. This representation is often realized as a latent, vectorized encoding within a neural network, though it may take other forms [7, 144, 228, 281]. A common example of concept representation is an embedding. In music generation, for instance, when models are trained on music paired with control attributes (e.g., genre, mood, or instrumentation), these attributes can be viewed as concepts, and their corresponding embeddings serve as the internal representations of these concepts. These embeddings are meaningful in that they can be manipulated to control specific aspects of the generated music [160, 205].

The real challenge in concept representation lies in capturing *implicit concepts*, such as pitch contour, texture, or timbre, as illustrated by the dashed boxes in Figure 1.1. These concepts cannot be precisely defined or directly labeled, making it difficult to construct explicit objective functions for learning them. For instance, pitch contour might be approximated using features such as pitch range, interval sequences, or frequency-domain coefficients [53], yet it ultimately remains something intuitively perceived rather than strictly quantifiable. From a cognitive science perspective, these concepts are rooted in perceptual experience and often shaped through multiple sensory modalities, making them subtle and difficult to define in symbolic terms [12]. Yet concepts like pitch contour are considered basic and universal aspects of musical understanding, recognizable even by infants before they acquire language or formal musical knowledge [237].

The ability to represent concepts gives rise to a special capability of *analogy-making*. For example, consider the task of generating a new melody that preserves the pitch contour but applies a different rhythm. Without an internal representation of pitch contour as a distinct concept, a model cannot preserve it. This contour-preserving transformation is also challenging to approach in a supervised manner, as it is nearly impossible to curate large datasets of paired melodies where the contour remains the same while other aspects, such as rhythm, vary in controlled ways.

The general methodology I use is called *disentanglement*, which involves encouraging specific parts of a representation to correspond to ideal concepts by regularizing the neural network with appropriate inductive biases. In Chapters 2–4, we explore disentanglement primarily within a variational autoencoder framework. Chapter 2 focuses on disentangling pitch contour and rhythmic pattern from 8-beat melodies. Chapter 3 extends this idea to disentangle chord and texture from 8-beat piano accompaniments. Chapter 4 investigates learning texture representations from both audio and symbolic data, applying them to symbolic arrangement. In contrast, Chapter 5 presents disentanglement in the context of fine-tuning a modern multimodal language model, where the goal is to learn accompaniment style representations from audio for symbolic arrangement.

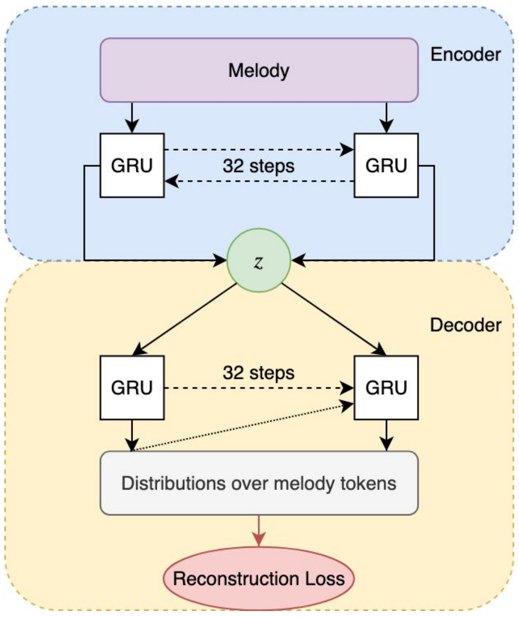
In this chapter, we begin with the problem of disentangling pitch and rhythm in monophonic melodies and introduce the perspective on analogy-making. This chapter is based on the published work *Deep Music Analogy Via Latent Representation Disentanglement* [271], conducted in collaboration with Ruihan Yang, Dingsu Wang, Tianyao Chen, Junyan Jiang, and Gus Xia.

2.1 Analogy-Making and Disentanglement

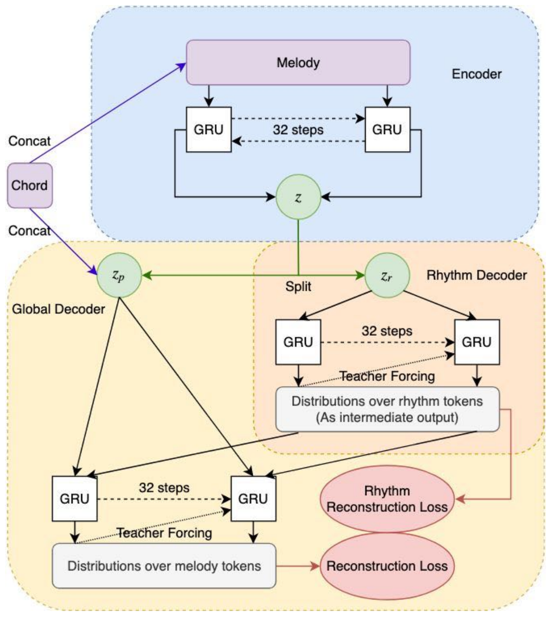
For intelligent systems, an effective way to generate high-quality art is to produce analogous versions of existing examples [97]. In general, two systems are analogous if they share common abstractions, i.e., high-level representations and their relationships, which can be revealed by the paired tuples *A : B :: C : D* (often spoken as A is to B as C is to D). For example, the analogy "the hydrogen atom is like our solar system" can be formatted as *Nucleus : Hydrogen atom :: Sun : Solar system*, in which the shared abstraction is "a bigger part is the center of the whole system." For generative algorithms, a clever shortcut is to make analogies by solving the problem of "*A : B :: C : ?*". In the context of music generation, if A is the rhythm pattern of a very lyrical piece B, this analogy can help us realize the imaginary situation of "what if B is composed with a rather rapid and syncopated rhythm C" by preserving the pitch contours and the intrinsic relationship between pitch and rhythm. In the same fashion, other types of "what if" compositions can be created by simply substituting A and C with different aspects of music (e.g., chords, melody, etc.).

A great advantage of *generation via analogy* is the ability to produce both *natural* and *creative* results. Naturalness is achieved by reusing the representations (high-level concepts such as "image style" and "music pitch contour") of human-made examples and the intrinsic relationship between the concepts, while creativity is achieved by recombining the representations in a novel way. However, making meaningful analogies also requires *disentangling* the representations, which is effortless for humans but non-trivial for computers. We already see that making analogies is essentially transferring the abstractions, not the observations—simply copying the notes or samples from one piece to another would only produce a casual re-mix, not an analogous composition [74].

In this chapter, we contribute an explicitly-constrained conditional variational autoencoder (EC^2-VAE), a conditional VAE with explicit semantic constraints on intermediate outputs of the network, as an effective tool for learning disentanglement. To be specific, the encoder extracts



(a) Vanilla sequence VAE.



(b) EC^2-VAE model.

Figure 2.1: A comparison between vanilla sequence VAE [272] and our model with condition and disentanglement.

latent representations from the observations; the semantic constraints disentangle the representations so that each part has a unique interpretation, and the decoder maps the disentangled representations back to actual music while preserving the intrinsic relationship between the representations. In producing analogies, we focus on disentangling and transferring the *pitch* and *rhythm* representations of 8-beat music clips when chords are given as the condition (an extra input) of the model. We show that EC^2-VAE has three desired properties as a generative model. First, the disentanglement is *explicitly coded*, i.e., we can specify which latent dimensions denote which semantic factors in the model structure. Second, the disentanglement does not sacrifice much of the reconstruction. Third, the learning does not require any analogous examples in the training phase, but the model is capable of making analogies in the inference phase. For evaluation, we propose a new metric and conduct a survey. Both objective and subjective evaluations show that our model significantly outperforms the baselines.

2.2 Methodology

In this section, we introduce the data representation and model design in detail. We focus on disentangling the latent representations of pitch and rhythm, the two fundamental aspects of composition, over the duration of 8-beat melodies. All data come from the Nottingham dataset [72], regarding a $\frac{1}{4}$ beat as the shortest unit.

2.2.1 Data Representation

Each 8-beat melody is represented as a sequence of 32 one-hot vectors, each with 130 dimensions, where each vector denotes a $\frac{1}{4}$-beat unit. As in [203], the first 128 dimensions denote the *onsets* of MIDI pitches ranging from 0 to 127 with one unit of duration. The 129^{th} dimension is the *holding* state for longer note duration, and the last dimension denotes *rest*. We also designed a rhythm feature to constrain the intermediate output of the network. Each 8-beat rhythm pattern is also represented as a sequence of 32 one-hot vectors. Each vector has 3 dimensions, denoting: an onset of any pitch, a holding state, and a rest.

Besides, chords are given as a condition, i.e., an extra input, of the model. The chord condition of each 8-beat melody is represented as a chromagram with equal length, i.e., 32 multi-hot vectors each with 12 dimensions, where each dimension indicates whether a pitch class is activated.

2.2.2 Model Architecture

Our model design is based on the previous studies of [203, 272], both of which used VAEs to learn the representations of fixed-length melodies. Figure 2.1 shows a comparison between the model architectures, where Figure 2.1(a) shows the model designed in [272] and Figure 2.1(b) shows the model design in this study. We see that both use bi-directional GRUs [48] (or LSTMs [100]) as the encoders (in blue) to map each melody observation to a latent representation z, and both use uni-directional GRUs (or LSTMs) (with teacher forcing [234] in the training phrase) as the decoders (in yellow) to reconstruct melodies from z.

The key innovation of our model design is to assign a part of the decoder (in orange) with a specific subtask: to disentangle the latent rhythm representation z_r from the overall z by explicitly encouraging the intermediate output of z_r to match the rhythm feature of the melody. The other part of z is therefore everything but rhythm and interpreted as the latent pitch representation, z_p. Note that this explicitly coded disentanglement technique is quite flexible — *we can use multiple subparts of the decoder to disentangle multiple semantically interpretable factors of z simultaneously* as long as the intermediate outputs of the corresponding latent factors can be defined, and the model shown in Figure 2.1(b) is the simplest case of this family.

It is also worth noting that the new model uses chords as a condition for both the encoder and decoder. The advantage of chord conditioning is to free z from storing chord-related information. In other words, the pitch information in z is "detrended" by the underlying chord for better encoding and reconstruction. The cost of this design is that we cannot learn a latent distribution of chord progressions.

Encoder A single layer bi-directional GRU with 32 time steps is used to model $Q_\theta(z|x, c)$, where x is the melody input, c is the chord condition, and z is the latent representation. Chord conditions are given by concatenating with the input at each time step.

Decoder The global decoder models $P_\phi(x|z, c)$ by multiple layers of GRUs, each with 32 steps. For disentanglement, z is splitted into two halves z_p and z_r($z = \mathrm{concat}[z_\mathrm{r}, z_\mathrm{p}]$), each being a 128-dimensional vector. As a subpart of the global decoder, the rhythm decoder models $P_{\phi_\mathrm{t}}(r(x)|z)$ by a single layer GRU, where $r(x)$ is the rhythm feature of the melody. Meanwhile, the rhythm is concatenated with z_p and chord condition as the input of the rest of the global decoder to reconstruct the melody. We used cross-entropy loss for both rhythm and melody reconstruction. Note that the overall decoder is supposed to learn non-trivial relationships between pitch and rhythm, rather than naively cutting a pitch contour by a rhythm pattern.

2.2.3 Theoretical Justification of the ELBO Objective with Disentanglement

One concern about representation disentanglement techniques is that they sometimes sacrifice reconstruction power [122]. In this section, we prove that our model does not suffer much of the disentanglement-reconstruction paradox, and the likelihood bound of our model is close to that of the original conditional VAE, and in some cases, equal to it.

Recall the Evidence Lower Bound (ELBO) objective function used by a typical conditional VAE [61] constraint on input sample x with condition c:

$$\text{ELBO}(\phi, \theta) = \mathbb{E}_Q[\log P_\phi(x|z, c)] \tag{2.1}$$

$$- \mathbb{KL}[Q_\theta(z|x, c)||P_\phi(z|c)] \leq \log P_\phi(x|c). \tag{2.2}$$

For simplicity, $\mathcal{D}$ denotes $\mathbb{KL}[Q_\theta(z|x, c)||P_\phi(z|c)]$ in the rest of this section. If we see the intermediate rhythm output in Figure 2.1(b) as hidden variables of the whole network, the new ELBO objective of our model only adds the rhythm reconstruction loss based on the original one, resulting in a lower bound of the original ELBO. Formally,

$$\begin{aligned} &\text{ELBO}^{\text{new}}(\phi, \theta) \\ =\ &\mathbb{E}_Q[\log P_\phi(x|z, c)] - \mathcal{D} + \mathbb{E}_Q[\log P_{\phi_{\text{t}}}(r(x)|z_{\text{r}})] \\ =\ &\text{ELBO}(\phi, \theta) + \mathbb{E}_Q[\log P_{\phi_{\text{t}}}(r(x)|z_{\text{r}})], \end{aligned}$$

where ϕ_{t} denotes parameters of the rhythm decoder. Clearly, ELBO^{new} is a lower bound of the original ELBO because $\mathbb{E}_Q[\log P_{\phi_{\text{t}}}(r(x)|z_{\text{r}})] \leq 0$.

Moreover, if the rest of global decoder takes the original rhythm rather than the intermediate output of rhythm decoder as the input, the objective can be rewritten as:

$$\begin{aligned}
&\text{ELBO}^{\text{new}}(\phi, \theta) \\
&= \mathbb{E}_Q[\underbrace{\log P_\phi(x|r(x), z_\text{p}, c) + \log P_\phi(r(x)|z_\text{r}, c)}_{\text{with } x \perp\!\!\!\perp z_\text{r}|r(x), c \text{ and } r(x) \perp\!\!\!\perp z_\text{p}|z_\text{r}, c}] - \mathcal{D} \\
&= \mathbb{E}_Q[\log P_\phi(x, r(x)|z, c)] - \mathcal{D} \\
&= \mathbb{E}_Q[\log P_\phi(x|z, c) + \log P_\phi(r(x)|x, z, c)] - \mathcal{D} \\
&= \text{ELBO}(\phi, \theta).
\end{aligned}$$

The second equal sign holds for a perfect disentanglement, and the last equal sign holds since $r(x)$ is decided by x, i.e., $P_\phi(r(x)|x, z, c) = 1$. In other words, we show that under certain assumptions ELBO$^{\text{new}}$ with disentanglement is identical to the ELBO.

2.3 Experiments

We present the objective metrics to evaluate the disentanglement in Section 2.3.1, show several representative examples of generation via analogy in Section 2.3.2, and use subjective evaluations to rate the artistic aspects of the generated music in Section 2.3.3.

2.3.1 Objective Measurements

Upon a successful pitch-rhythm disentanglement, any changes in pitch of the original melody should not affect the latent rhythm representation much, and vice versa. Following this assumption, we developed two measurements to evaluate the disentanglement: 1) Δz after transposition, which is more qualitative, and 2) F-score of an augmentation-based query, which is more quantitative.

Visualizing Δz After Transposition

We define F_i as the operation of transposing all the notes by i semitones, and use the L_1-norm to measure the change in z. Figure 2.2 shows a comparison between $\Sigma|\Delta z_{\mathrm{p}}|$ and $\Sigma|\Delta z_{\mathrm{r}}|$ when we apply F_i to a randomly chosen piece (where $i \in [1, 12]$) while keeping the rhythm and underlying chord unchanged.

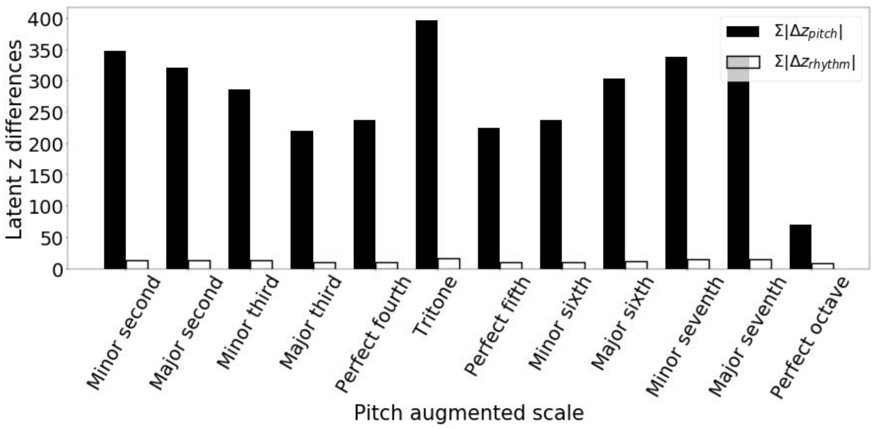


Figure 2.2: A comparison between Δz_{p} and Δz_{r} after transposition.

Here, the black bars stand for $\Sigma|\Delta z_{\mathrm{p}}|$ and the white bars stand for the $\Sigma|\Delta z_{\mathrm{r}}|$. It is conspicuous that when augmenting pitch, the change of z_{p} is much larger than the change of z_{r}, which well demonstrates the success of the disentanglement.

It is also worth noting that the change of z_{p} to a certain extent *reflects human pitch perception*. Given a chord, the change in z_{p} can be understood as the "burden" (or difficulty) to memorize (or encode) a transposed melody. We see that such a burden is large for tritone (very dissonant), relatively small for major third, perfect fourth & fifth (consonant), and very small for perfect octave.

Due to the space limit, we only show the visualization of the latent space when changing the pitch. According to the data representation in Section 2.2.1, changing the rhythm feature of a melody would inevitably affect the pitch contour, which would lead to complex behavior of the latent space hard to interpret visually. We leave the discussion for future work, but will pay more attention to the effect of the rhythm factor in Section 2.3.3.

F-Score of Augmentation-Based Query

The explicitly coded disentanglement enables a new evaluation method from an *information-retrieval* perspective. We regard the pitch-rhythm split in z defined by the model structure as the *reference* (the ground truth), the operation of factor-wise data augmentation (keeping the rhythm and only changing pitch randomly, or vice versa) as a *query* in the latent space, and the actual latent dimensions having the largest variance caused by augmentation as the *result set*. In this way, we can quantitatively evaluate our model in terms of precision, recall, and F-score.

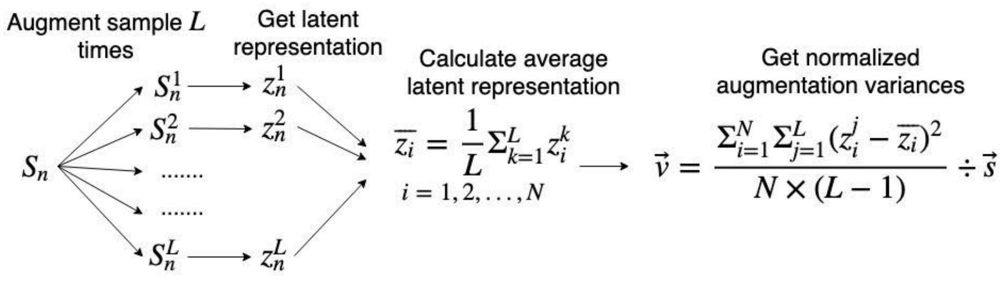

Augment sample L times
Get latent representation
Calculate average latent representation
Get normalized augmentation variances
S_n
S_n^1
S_n^2
S_n^L
z_n^1
z_n^2
z_n^L
$\overline{z_i} = \frac{1}{L}\Sigma_{k=1}^{L} z_i^k$
$i = 1, 2, \ldots, N$
$\vec{v} = \frac{\Sigma_{i=1}^{N}\Sigma_{j=1}^{L}(z_i^j - \overline{z_i})^2}{N \times (L-1)} \div \vec{s}$

Figure 2.3: Evaluating the disentanglement by data augmentation.

Figure 2.3 shows the detailed query procedure, which is a modification of the evaluation method in [122]. After pitch or rhythm augmentation for each sample, $\vec{v}$ is calculated as the average (across the samples) variance (across augmented versions) of the latent representations, normalized by the total sample variance $\vec{s}$. Then, we choose the first half (128 dimensions) with the largest variances as the result set. The precision, recall, and F-score of this augmentation-based query result are shown in Table 2.1. (Here, precision and recall are identical since the size of the result set equals the dimensionality of z_p and z_r.) As this is the first tailored metric for explicitly coded disentanglement, we use a random guess as our baseline.

	Pitch			Rhythm		
	pre.	rec.	F-s.	pre.	rec.	F-s.
EC^2-VAE	**0.88**	**0.88**	**0.88**	**0.80**	**0.80**	**0.80**
Random	0.5	0.5	0.5	0.5	0.5	0.5

Table 2.1: The evaluation results of pitch- and rhythm-wise augmentation-based query.

2.3.2 Examples of Generation via Analogy

We present several representative“what if” examples by swapping or interpolating the latent representations of different pieces. Throughout this section, we use the following example (shown in Figure 2.4), an 8-beat melody from the Nottingham Dataset [72] as the source, and the target rhythm or pitch will be borrowed from other pieces.[1]

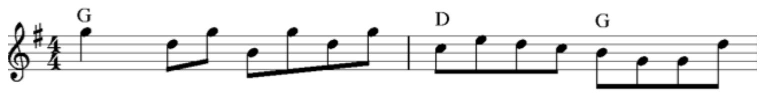
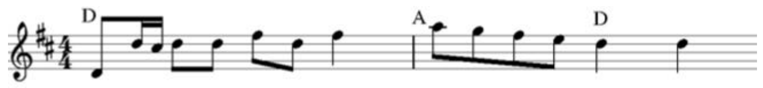


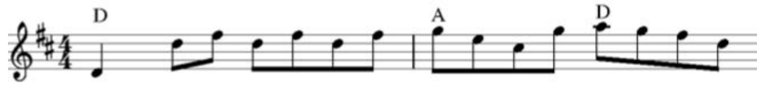
Figure 2.4: The source melody.

Analogy by Replacing z_p

Two examples are presented. In both cases, the latent pitch representation and the chord condition of the source melody are replaced with new ones from other pieces. In other words, the model answers the analogy question: *“source’s pitch : source melody :: target’s pitch : ?”*



(a) Target’s pitch and chord.

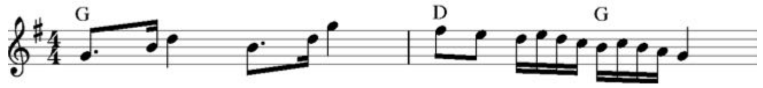


(b) The generated target music, using the pitch and chord from (a) and the rhythm from the source.

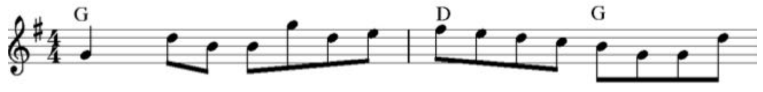
Figure 2.5: The first example of analogy via replacing z_p.

[1]MIDI demos are available at https://github.com/cdyrhjohn/Deep-Music-Analogy-Demos.

Figure 2.5 shows the first example, where Figure 2.5(a) shows the piece from which the pitch and chords are borrowed, and Figure 2.5(b) shows the generated melody. From Figure 2.5(a), we see the target melody is in a different key (D major) with a larger pitch range than the source and a big pitch jump in the beginning. From Figure 2.5(b), we see the generated new melody captures such pitch features while keeping the rhythm of the source unchanged.



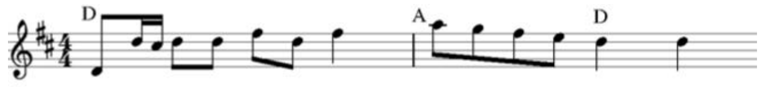
(a) Target's pitch and chord.



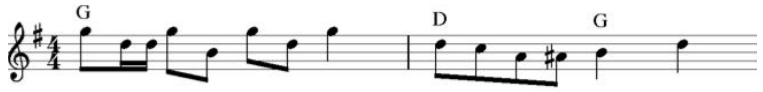
(b) The generated target, using the pitch and chord from (a) and the rhythm from the source.

Figure 2.6: The second analogy example via replacing z_{p}.

Figure 2.6 shows another example, whose subplots share the same meanings with the previous one. From Figure 2.6(a), we see the first measure of the target's melody is a broken chord of Gmaj, while the second measure is the G major scale. From Figure 2.6(b), we see the generated new melody captures these pitch features. Moreover, it retains the source's rhythm and ignores the dotted eighth and sixteenth notes in Figure 2.6(a).



(a) Target's rhythm pattern.

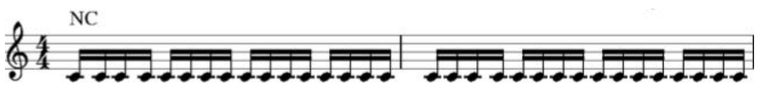


(b) The generated target music, using the rhythm of (a) while keeping the source's pitch and chord.

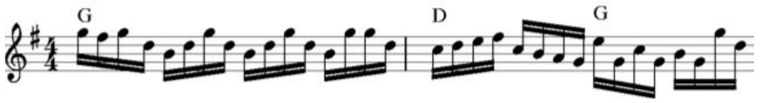
Figure 2.7: The first example of analogy via replacing z_{r}.

Analogy by Replacing z_{r}

Similar to the previous section, this section shows two example answers to the question:"*source's rhythm : source melody :: target's rhythm : ?*" by replacing z_{r}. Figure 2.7 shows the first example, where Figure 2.7(a) contains the new rhythm pattern quite different from the source, and Figure 2.7(b) is the generated target. We see that Figure 2.7(b) perfectly inherited the new rhythm pattern and made minor but novel modifications based on the source's pitch.



(a) Target's rhythm pattern.

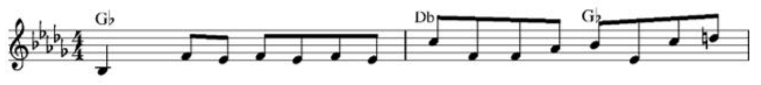


(b) The generated target music, using the rhythm of (a) while keeping the source's pitch and chord.

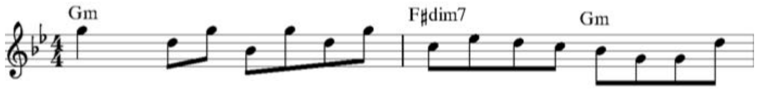
Figure 2.8: The second analogy example via replacing z_{r}.

Figure 2.8 shows a more extreme case, in which Figure 2.8(a) contains only 16th notes of the same pitch. Again, we see the generated target in Figure 2.8(b) maintains the source's pitch contour while matching the given rhythm pattern.

Analogy by Replacing Chord



(a) Changing all the chords down a semitone, resulting in the key change from G major to Bb minor.



(b) Changing the key from G major to G minor.

Figure 2.9: Two examples of replacing the original chord.

Though chord is not our main focus, here we show two analogy examples in Figure 2.9 to answer "what if" the source melody is composed using some other chord progressions. Figure 2.9(a) shows an example where the key is B♭ minor. An interesting observation is that the new melody contour indeed adds some reasonable modification (e.g., flipping the melody) rather than simply transposing down all the notes. It brings us a little sense of Jazz. Figure 2.9(b) shows an example where the key is changed from G major to G minor. We see melody also naturally transforms from major mode to minor mode.

Two-Way Pitch-Rhythm Interpolation

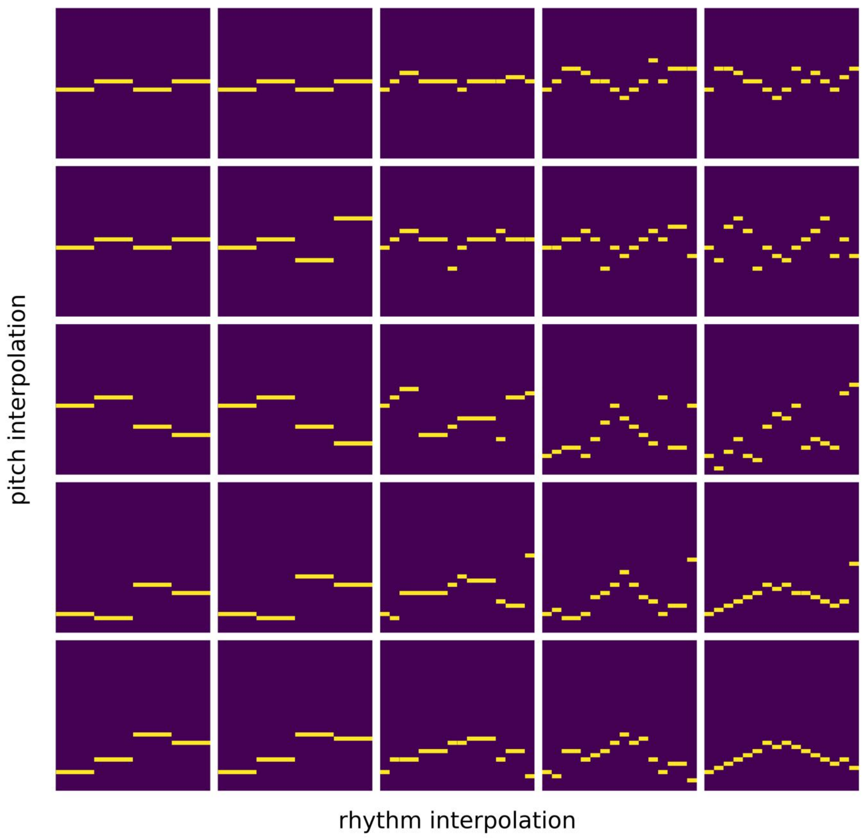


Figure 2.10: An illustration of two-way interpolation.

The disentanglement also enables a smooth transition from one music to another. Figure 2.10 shows an example of two-way interpolation, i.e., a traversal over a subspace of the learned latent representations z_{r} and z_{p} along two axes respectively, while keeping the chord as NC (no chord). Here, each square is a piano-roll of an 8-beat music. The top-left (source) and bottom-right (target) squares are two samples created manually, and everything else is generated by interpolation using SLERP [251]. Note that the rhythmic changes are primarily observed moving along the "rhythm interpolation" axis, and likewise for pitch and the vertical "pitch interpolation" axis.

2.3.3 Subjective Evaluation

Besides objective measurement, we conducted a subjective survey to evaluate the quality of generation via analogy. We focus on changing the rhythm factors of existing music since this operation leads to an easier identification of the source melodies.

Each subject listened to two groups of five pieces each. All the pieces had the same length (64 beats at 120 bpm). Within each group, one piece was an original, human-composed piece from the Nottingham dataset, having a lyrical melody consisting of longer notes. The remaining four pieces were variations upon the original with more rapid rhythms consisting of 8^{th} and 16^{th} notes. Two of the variations were produced in a rule-based fashion by naively cutting the notes in the original into shorter subdivisions, serving as the *baseline*. The other two variations were generated with our EC^2-VAE by merging the z_{p} of the original piece and the z_{r} decoded from two pieces having the same rhythm pattern as the baselines but with all notes replaced with C4 (similar to Figure 2.8(a). The subjects always listened to the original first, and the order of the variations was randomized. In sum, we compare three versions of music: 1) the original piece, 2) the variation created by the baseline, and, 3) the variation created by our algorithm. The subjects were asked to rate each sample on a 5-point scale from 1 (very low) to 5 (very high) according to three criteria:

1. *Creativity*: how creative the composition is.
2. *Naturalness*: how human-like the composition is.
3. *Overall musicality.*

A total of 30 subjects (16 female and 14 male) participated in the survey. Figure 2.11 shows the results, where the heights of bars represent means of the ratings the and error bars represent the MSEs computed via within-subject ANOVA [208]. The result shows that our model performs significantly better than the rule-based baseline in terms of creativity and musicality ($p < 0.05$), and marginally better in terms of naturalness. Our proposed method is even comparable to the original music in terms of creativity, but remains behind human composition in terms of the other two criteria.

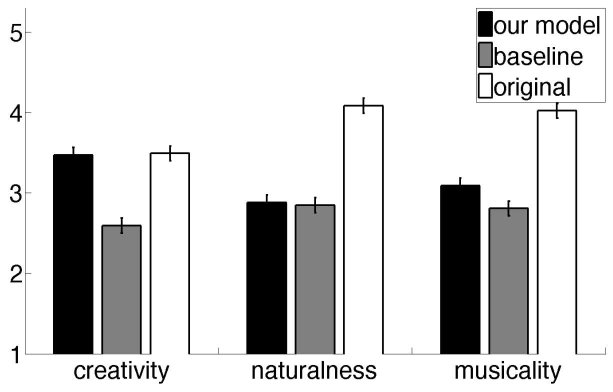


Figure 2.11: Subjective evaluation results.

2.4 Related Work

2.4.1 Generation Via Analogy

The history of generation via analogy can trace back to the studies of non-parametric "image analogies" [97] and "playing Mozart by analogy" using case-based reasoning [257]. With recent breakthroughs in artificial neural networks, we see a leap in the quality of produced analogous examples using deep generative models, including music and image style transfer [56, 77], image-to-image translation [111], attribute arithmetic [32], and voice impersonation [73].

Here, we distinguish between two types of analogy algorithms. In a *broad* sense, an analogy algorithm is any computational method capable of producing analogous versions of existing examples. A common and relatively easy approach is supervised learning, i.e., to directly learn the mapping between analogous items from labeled examples [111, 236]. This approach requires little representation learning but needs a lot of labeling effort. Moreover, supervised analogy does not generalize well. For example, if the training analogous examples are all between lyrical melodies (the source domain) and syncopated melodies (the target domain), it would be difficult to create other rhythmic patterns, much less the manipulation of pitch contours. (Though improvements [22, 123, 290] have been made, weak supervision is still needed to specify the source and target domains.) On the other hand, a *strict* analogy algorithm requires not only learning the representations but also disentangling them, which would allow the model to make domain-free analogies via the manipulation of any disentangled representations. Our approach belongs to this type.

2.4.2 Representation Learning and Disentanglement

Variational auto-encoders (VAEs) [125] and generative adversarial networks (GANs) [78] are so far the two most popular frameworks for music representation learning. Both use encoders (or discriminators) and decoders (or generators) to build a bi-directional mapping between the distributions of observation x and latent representation z, and both generate new data via sampling from $p(z)$. For music representations, VAEs [30, 69, 203, 272] have been a more successful tool so far compared with GANs [274], and our model is based on the previous study [272].

The motivation of representation disentanglement is to better interpret the latent space generated by VAE, connecting certain parts of z to semantic factors (e.g., age for face images, or rhythm for melody), which would enable a more controllable and interactive generation process. InfoGAN [39] disentangles z by encouraging the mutual information between x and a subset of z. β-VAE [98] and its follow-up studies [38, 122, 272] imposed various extra constraints and properties on $p(z)$. However, the disentanglement above is still *implicit*, i.e., though the model separates the latent space into subparts, we cannot define their meanings beforehand and have to "check it out" via *latent space traversal* [32]. In contrast, the disentanglement in Style-based GAN [120], Disentangled Sequential Autoencoder [147], and our EC^2-VAE are *explicit*, i.e., the meanings of different parts of z are defined by the model structure, so that the controlled generation is more precise and straightforward. The study Disentangled Sequential Autoencoder [147] is most related to our work and also deals with sequential inputs. Using a partially time-invariant encoder, it can approximately disentangle dynamic and static representations. Our model does not directly constrain z but applies a loss to intermediate outputs associated with latent factors. Such an indirect but explicit constraint enables the model to further disentangle the representation into pitch, rhythm, and any semantic factors whose observation loss can be defined. As far as we know, this is the first disentanglement learning method tailored for music composition.

2.5 Summary

In summary, we contributed an explicitly-constrained conditional variational autoencoder (EC^2-VAE) as an effective disentanglement learning model. This model generates new music via making analogies, i.e., to answer the imaginary situation of "what if" a piece is composed using different pitch contours, rhythm patterns, and chord progressions via replacing or interpolating the disentangled representations. Experimental results showed that the disentanglement is successful and the model is able to generate interesting and musical analogous versions of existing music. We see this study as a significant step in music understanding and controlled music generation. The model can also be generalized to other tasks and domains, shedding light on the general scenario of generation via analogy.

3 | Disentangling Chord and Texture in Polyphonic Accompaniment

In this chapter, we address the problem of concept learning in polyphonic music. Building on the idea of representation disentanglement and analogy-making introduced in Chapter 2, we propose a novel architecture that learns two interpretable latent factors from polyphonic piano music: *chord* and *texture*. The focus of this chapter is on the controllable generation capabilities enabled by these disentangled representations.

This chapter is based on the published work *Learning Interpretable Representation for Controllable Polyphonic Music Generation* [247], conducted in collaboration with Dingsu Wang, Yixiao Zhang, and Gus Xia.[1] This work also builds upon prior research in representation learning for polyphonic music, the PianoTree VAE with a hierarchical encoding and decoding structure, which is introduced in Appendix A.

3.1 Disentanglement for Controllable Music Generation

With the development of artificial neural networks, deep learning has become one of the most popular techniques for automated music generation. In particular, we see recurrent and attention-based models being able to generate creative and human-like music without heavily handcrafted

[1]Code and demos are available at https://github.com/ZZWaang/polyphonic-chord-texture-disentanglement.

rules [37, 106, 108]. However, the main drawback of these deep generative models is that they behave like "black boxes", and it is difficult to interpret the musical meaning of their internal latent variables [28]. Consequently, it remains a challenging task to control the generation process (i.e., to guide the music flow by manipulating the high-level compositional factors such as melody contour, accompaniment texture, style, etc.). This limitation restricts the application scenario of the powerful deep generative models.

In this chapter, we improve the model interpretability for music generation via constrained representation learning. Inspired by the content-style disentanglement idea [56], we enforce the model to learn two fundamental factors of polyphonic music: *chord* (content) and *texture* (style). The former refers to the representation of the underlying chord progression, and the latter includes chord arrangement, rhythmic pattern, and melody contour. The current design focuses on learning 8-beat long piano composition segments under a variational autoencoder (VAE) framework.

The core of the model design lies in the encoder. We incorporate the encoder with two inductive biases for a successful *chord-texture disentanglement*. The former applies a rule-based chord recognizer and embeds the information into the first half of the latent representation. The latter regards music as 2-D images and uses a chord-invariant convolutional network to extract the texture information, storing it in the second half of the latent representation. As for the decoder, we adopt the design from PianoTree VAE [248], an architecture that can reconstruct polyphonic music from the latent representation in a hierarchical manner.

We further show that the interpretable representations are *general-purpose*, empowering a wide spectrum of controllable music generation. In this study, we explore the following three scenarios:

1. **Task 1: Compositional style transfer** by swapping the chord and texture factors of different pieces of music, which can help us re-harmonize or re-arrange a music piece following the style of another piece.

2. **Task 2: Texture variation** by sampling the texture factor while keeping the chord factor, which is analogous to the creation of "theme and variations" form of composition.

3. **Task 3: Accompaniment arrangement** by predicting the texture factor given the melody using a downstream encoder-decoder generative model.

In sum, the contributions of the chapter are as follows:

1. We design a representation disentanglement method for polyphonic music, which learns two interpretable factors: chord and texture.

2. We show that the interpretable factors are general-purpose features for controllable music generation, which reduces the necessity to design heavily engineered control-specific model architectures. As far as we know, this is the first attempt to explicitly control the compositional texture feature for symbolic polyphonic music generation.

3. We demonstrate that control methods are effective and the quality of generated music is high. Some style-transferred pieces are rated even higher than the original ones composed by humans.

3.2 Model

In this section, we introduce the model design and data representation in detail. The goal is to learn the representations of 8-beat long piano compositions (with $\frac{1}{4}$ beat as the shortest unit) and disentangle the representations into two interpretable factors: chord and texture.

Figure 3.1 shows the overall architecture of the model. It adopts a VAE framework and contains four parts: 1) a chord encoder, 2) a chord decoder, 3) a texture encoder, and 4) a PianoTree decoder. The chord encoder and chord decoder can be seen as a standalone VAE, which extracts the latent chord representation z_{chd}. On the other hand, the texture encoder aims to extract the

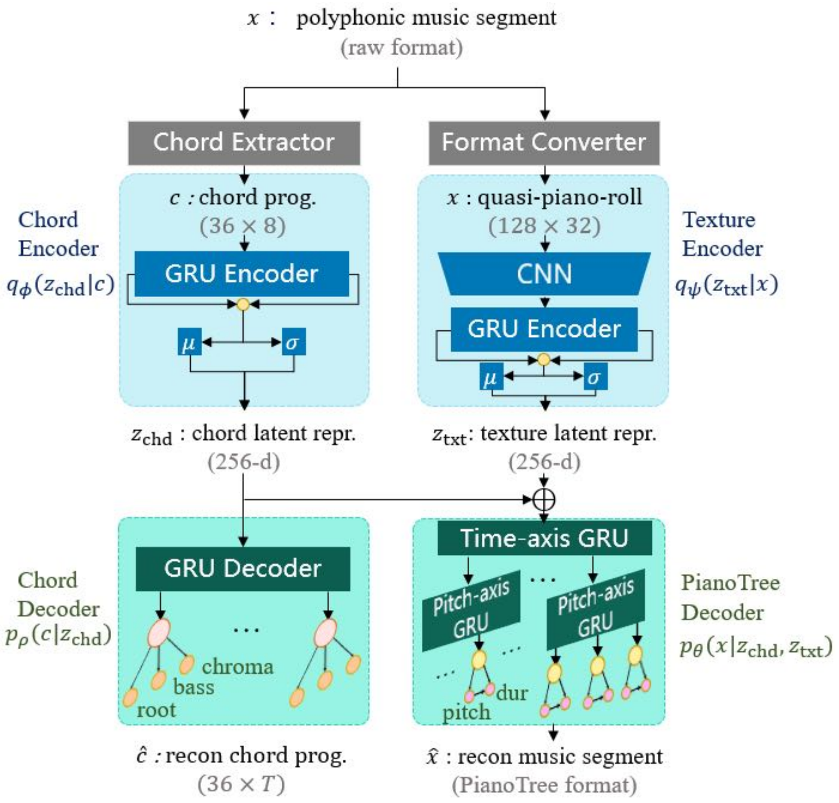


Figure 3.1: The model diagram of polyphonic disentanglement.

texture representation z_{txt} using a chord-invariant convolutional mapping. Finally, the PianoTree decoder takes in both z_{chd} and z_{txt} and outputs the original music in a tree-structured data format.

3.2.1 Chord Encoder

The chord encoder first applies rule-based methods [190, 201] to extract the chord progression under one-beat resolution. Each extracted chord progression is a 36×8 matrix, where each column denotes a chord of one beat. Each chord is a 36-D vector consisting of three parts: a 12-D one-hot vector for the pitch class of the *root*, a 12-D one-hot vector for the *bass*, and a 12-D multi-hot *chroma* vector.

The chord progression is then fed into a bi-directional GRU encoder [203], and the last hidden states on both ends of the GRU are concatenated and used to approximate the posterior distribution of z_{chd}. Following the assumption of a standard VAE, z_{chd} has a standard Gaussian prior and follows an isotropic Gaussian posterior.

Note that although the chord progression here is extracted using algorithms, it can also be provided by external labels, in which case the whole model becomes a conditional VAE [269].

3.2.2 Chord Decoder

The chord decoder reconstructs the chord progression from z_{chd} using another bi-directional GRU. The reconstruction loss of a chord progression is computed as a summation of 8 beat-wise chord loss using cross entropy functions [19]. For each beat, the chord loss is defined as the product of three parts: 1) the root loss, 2) the bass loss, and 3) the chroma loss. The root and bass are both considered 12-way categorical distributions, and the chroma is regarded as 12 independent Bernoulli distributions.

3.2.3 Texture Encoder

The input of the texture encoder is an 8-beat segment of a polyphonic piece represented by an image-like data format slightly modified from the piano-roll [65]. Each 8-beat segment is represented by a 128×32 matrix, where each row corresponds to a MIDI pitch and each column corresponds to $\frac{1}{4}$ beat. The data entry at (p, t) records the duration of the note if there is a note onset, and zero otherwise.

The texture encoder aims to learn a chord-invariant representation of texture by leveraging both the translation invariance property of convolution and the blurry effect of max-pooling layers [133]. We use a convolutional layer with kernel size 12×4 and stride 1×4, which is followed by a ReLU activation [177] and max-pooling with kernel size 4×1 and stride 4×1. The convolutional layer has one input channel and 10 output channels. The convolutional layer design aims at extracting a blurry "concept sketch" of the polyphonic texture, which contains minimum information about the underlying chord. Ideally, when the blurry sketch is combined with the chord representation, the decoder can identify its concrete pitches in a musical way.

The output of the convolutional layer is then fed into a bi-directional GRU encoder to extract the texture representation z_{txt}, similar to how we encode z_{chd} introduced in Section 3.2.1.

3.2.4 PianoTree Decoder

The PianoTree decoder takes the concatenation of z_{chd} and z_{txt} as input and decodes the music segment using the same decoder structure invented in PianoTree VAE [248], a hierarchical model structure for polyphonic representation learning. The decoder works as follows. First, it generates 32 frame-wise hidden states (one for each $\frac{1}{4}$ beat) using a GRU layer. Then, each frame-wise hidden state is further decoded into the embeddings of individual notes using another GRU layer. Finally, the pitch and duration for each note are reconstructed from the note embedding using a fully-connected layer and a GRU layer, respectively. For more detailed derivation and model design, we refer the readers to [248].

3.2.5 Training Objective

Let x denote the input music piece and $c = f(x)$ denote the chord progression extracted by algorithm $f(\cdot)$. We assume standard Gaussian priors of $p(z_{\text{chd}})$ and $p(z_{\text{txt}})$, and denote the output posteriors of chord encoder and texture encoder by $q_\phi(z_{\text{chd}}|c)$, $q_\psi(z_{\text{txt}}|x)$, the output distributions of chord decoder and PianoTree decoder by $p_\rho(c|z_{\text{chd}})$ and $p_\theta(x|z_{\text{chd}}, z_{\text{txt}})$. The objective of the model is:

$$\begin{aligned}
&\mathcal{L}(\phi, \psi, \rho, \theta; x) = \\
&-\mathbb{E}_{\substack{z_{\text{chd}} \sim q_\phi \\ z_{\text{txt}} \sim q_\psi}}\left[\log p_\rho(c|z_{\text{chd}}) + \log p_\theta(x|z_{\text{chd}}, z_{\text{txt}})\right] \\
&+ \mathrm{KL}(q_\phi || p(z_{\text{chd}})) + \mathrm{KL}(q_\psi || p(z_{\text{txt}})).
\end{aligned}$$

3.3 Controlled Music Generation

In this section, we show some controlled generation examples of the three tasks mentioned in the introduction.

3.3.1 Compositional Style Transfer

By regarding chord progression *content* and texture *style*, we can achieve compositional style transfer by swapping the texture representations of different pieces. Figure 3.2 shows the transferred results ((c) & (d)) based on two 16-bar samples ((a) & (b)) in the test set by swapping z_{txt} every 2 bars (without overlap).[2]

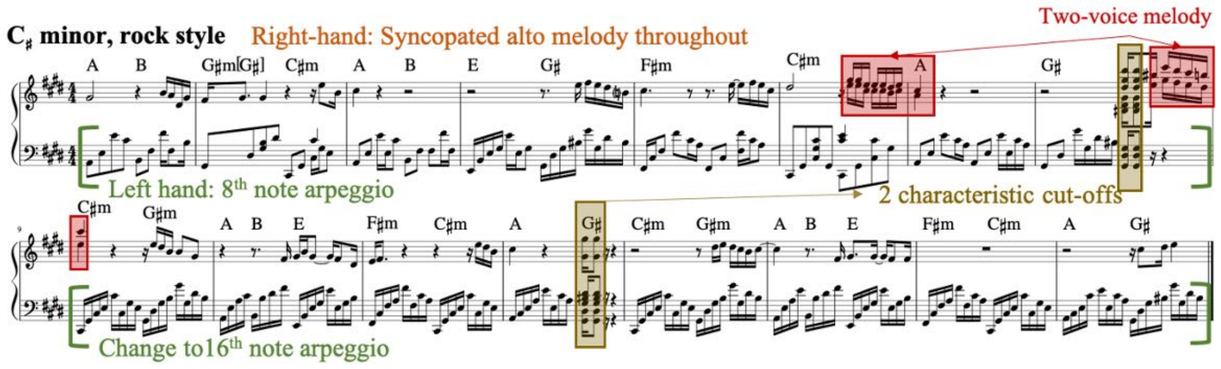
We see that such long-term style transfer is successful: The generated segment (c) follows the chord progression of (b) while mimicking the texture of (a), while (d) follows the chord progression of (a) while mimicking the texture of (b). As shown in the marked scores, the style transfer is effective. For example, the cut-offs, melody contours, and the shape of the left-hand accompaniment are all preserved.

3.3.2 Texture Variation by Sampling

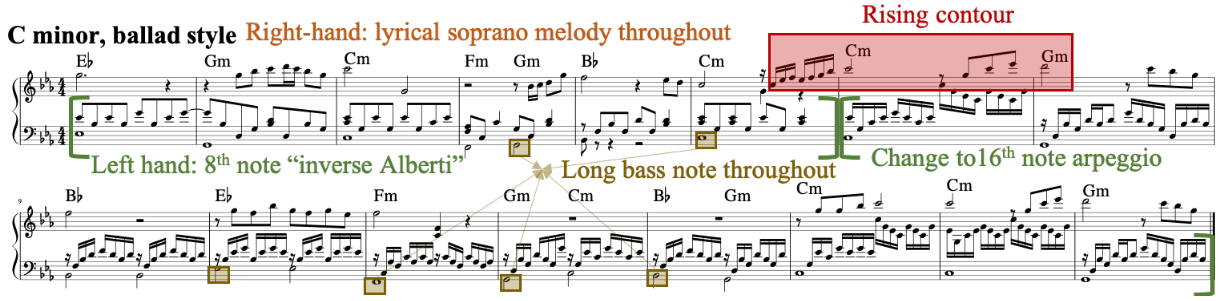
We can make variations of texture by sampling from z_{txt} while keeping z_{chd}. Here, we investigate two sampling strategies: sampling from the posterior $q_\psi(z_{\text{txt}}|x)$, and sampling from the prior $p(z_{\text{txt}})$.

Sampling from the posterior distribution $q_\psi(z_{\text{txt}}|x)$ yields reasonable variations as shown in Figure 3.3(a). The variations of the right-hand melody can be seen as an improvisation following the chord progression and the melody. On the contrary, there is only a small variation in the left-hand part, showing that the model regards the left-hand accompaniment as the dominant

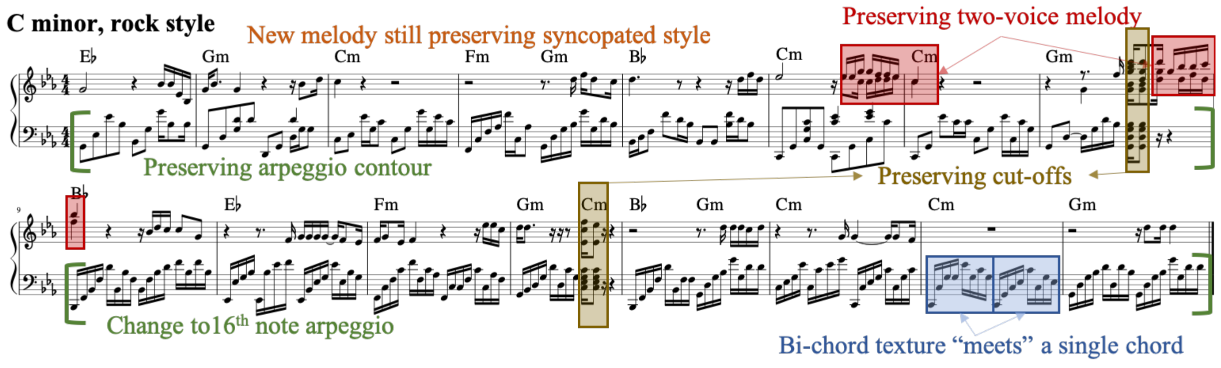
[2]The presented excerpts are converted from MIDI by the authors. The chord labels are inferred from the original/generated samples.


C♯ minor, rock style
Right-hand: Syncopated alto melody throughout
Two-voice melody
Left hand: 8th note arpeggio
2 characteristic cut-offs
Change to16th note arpeggio
A B G♯m[G♯] C♯m A B E G♯ F♯m C♯m A G♯
C♯m G♯m A B E F♯m C♯m A G♯ C♯m G♯m A B E F♯m C♯m A G♯

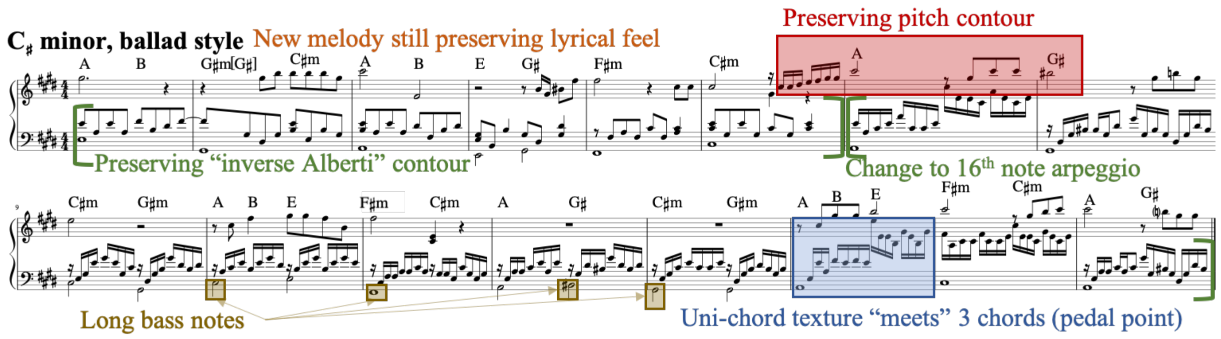
(a) A real piece.


C minor, ballad style
Right-hand: lyrical soprano melody throughout
Rising contour
Left hand: 8th note "inverse Alberti"
Long bass note throughout
Change to16th note arpeggio
E♭ Gm Cm Fm Gm B♭ Cm Cm Gm
B♭ E♭ Fm Gm Cm B♭ Gm Cm Cm Gm

(b) The other real piece.

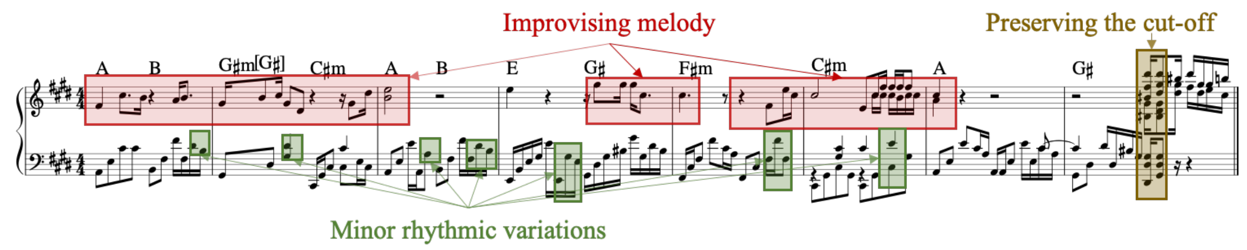

C minor, rock style
New melody still preserving syncopated style
Preserving two-voice melody
Preserving arpeggio contour
Preserving cut-offs
Change to16th note arpeggio
Bi-chord texture "meets" a single chord
E♭ Gm Cm Fm Gm B♭ Cm Cm Gm
B♭ E♭ Fm Gm Cm B♭ Gm Cm Cm Gm

(c) The generated piece by combining z_{txt} from (a) and z_{chd} from (b).

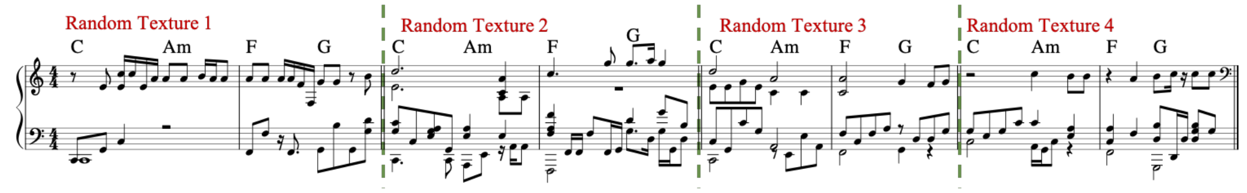

C♯ minor, ballad style
New melody still preserving lyrical feel
Preserving pitch contour
Preserving "inverse Alberti" contour
Change to 16th note arpeggio
Long bass notes
Uni-chord texture "meets" 3 chords (pedal point)
A B G♯m[G♯] C♯m A B E G♯ F♯m C♯m A G♯
C♯m G♯m A B E F♯m C♯m A G♯ C♯m G♯m A B E F♯m C♯m A G♯

(d) The generated piece by combining z_{txt} from (b) and z_{chd} from (a).

Figure 3.2: An example of compositional style transfer of 16-bar-long samples.



(a) An example of posterior sampling of z_{txt} of the first 8 bars of the segment (a) in Figure 3.2



(b) An example of prior sampling of z_{txt} under given chord progression C-Am-F-G. Each two-bar segment is independently sampled, having different texture.

Figure 3.3: Examples of texture variations via posterior sampling and prior sampling.

feature of texture.

Sampling from the prior distribution $p(z_{\text{txt}})$ changes the texture completely. Figure 3.3(b) shows a series of examples of prior sampling under the same chord progression C-Am-F-G. The resulting generations follow exactly the chord progression but with new textures.

3.3.3 Accompaniment Arrangement

We use a downstream predictive model to achieve accompaniment arrangement. For this task, we provide extra vocal melody tracks paired with the piano samples, and the model learns to generate 16-bar piano accompaniment *conditioned* on melody in a supervised fashion.

We encode the music every 2 bars (without overlap) into latent representations. For the accompaniment, we use the proposed model to compute the latent chord and texture representation, denoted by $\mathbf{z}_{\text{chd}} = [z_{\text{chd}}^{(1)}, ..., z_{\text{chd}}^{(4)}]$ and $\mathbf{z}_{\text{txt}} = [z_{\text{txt}}^{(1)}, ..., z_{\text{txt}}^{(4)}]$. For the melody, we use the EC^2-VAE [271] to compute the latent pitch and rhythm representations, denoted by $\mathbf{z}_{\text{p}} = [z_{\text{p}}^{(1)}, ..., z_{\text{p}}^{(4)}]$ and $\mathbf{z}_{\text{r}} = [z_{\text{r}}^{(1)}, ..., z_{\text{r}}^{(4)}]$. Then, we adopt a vanilla transformer [241] to model $p(\mathbf{z}_{\text{txt}}, \mathbf{z}_{\text{chd}} | \mathbf{z}_{\text{p}}, \mathbf{z}_{\text{r}})$, in which the encoder takes in the condition and the decoder's input is a shifted right version

$[\mathbf{z}_{\text{chd}}, \mathbf{z}_{\text{txt}}]$. Both encoder and decoder inputs are incorporated with a *positional encoding* indicating the time positions and a learned *factor embedding* indicating the representation type (i.e., pitch, rhythm, chord, or texture).

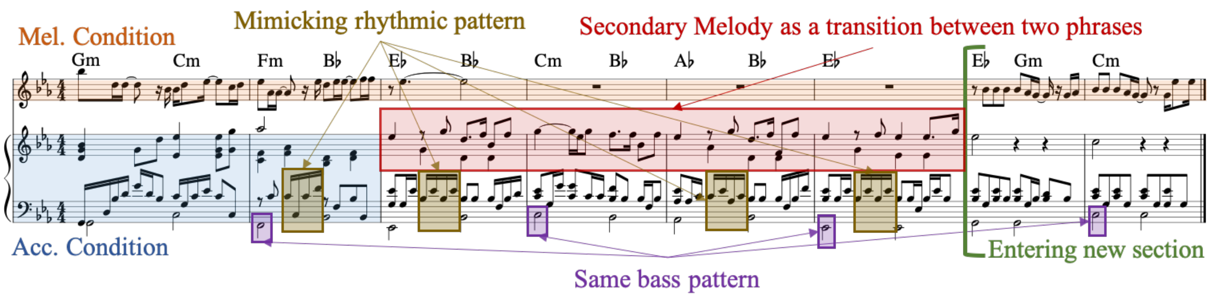


Figure 3.4: An example of accompaniment arrangement conditioned on melody, chord progression, and first 2 bars of accompaniment.

Figure 3.4 shows an example of accompaniment arrangement, where the first staff shows the melody and the second staff shows the piano accompaniment. In this case, the whole melody, together with the complete chord progression and the first two bars of the accompaniment, is given. The chord conditioning is achieved by forcing the decoded chord representation to match the given input during inference time. (A similar method is used in [64].) From Figure 3.4, we see that the model predicts a similar texture to the given accompaniment. Moreover, it fills in a secondary melody line as a transition when the lead melody is at rest.

Note that the arrangement can be generated in a flexible way by conditioning on different sets of latent factors.

3.4 Experiments

3.4.1 Dataset and Training

We train our model on the POP909 dataset [250], which contains about 1K MIDI files of pop songs (including paired vocal melody and piano accompaniment). We further extract the chord annotations using [190, 201]. We only keep the pieces with $\frac{2}{4}$ and $\frac{4}{4}$ meters and cut them into 8-beat music segments (so that each data sample in our experiment contains 32 time steps under 16th note resolution). In all, we have 66K samples. We randomly split the dataset (at song-level) into the training set (90%) and the test set (10%). All training samples are further augmented by transposing to all 12 keys.

In our experiment, the VAE model uses 256, 512, and 512 hidden dimensions for the GRUs in the chord encoder, the chord decoder, and the texture encoder, respectively. The latent dimension of z_{chd} and z_{txt} are both 256. The model size of the PianoTree decoder is the same as the implementation in the original paper [248]. The transformer model has the following size: hidden dimension $= 256$, number of layers $= 4$, and number of heads $= 8$.

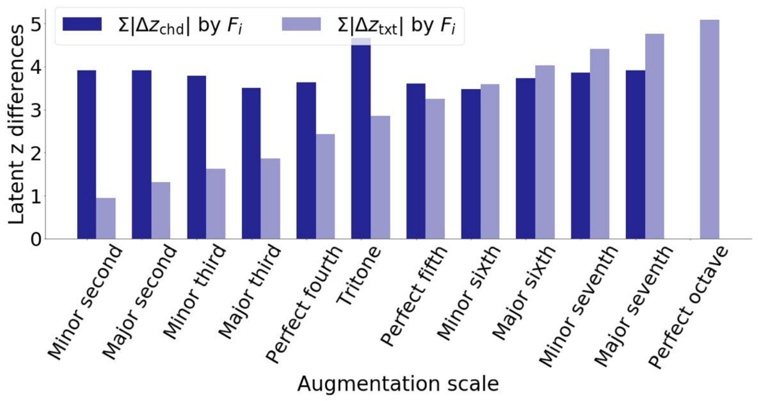
For both models, we use Adam optimizer [124] with a scheduled learning rate from 1e-3 to 1e-5. Moreover, for the VAE model, we use KL-annealing [25], i.e., setting a weight parameter for the KL-divergence loss starting from 0 to 0.1. We set the batch size to be 128, and the training converges within 6 epochs. For the downstream transformer model, we use 12K warmup steps for learning rate update [267]. We use the same batch size, and the model converges within 40 epochs.

3.4.2 Objective Measurement

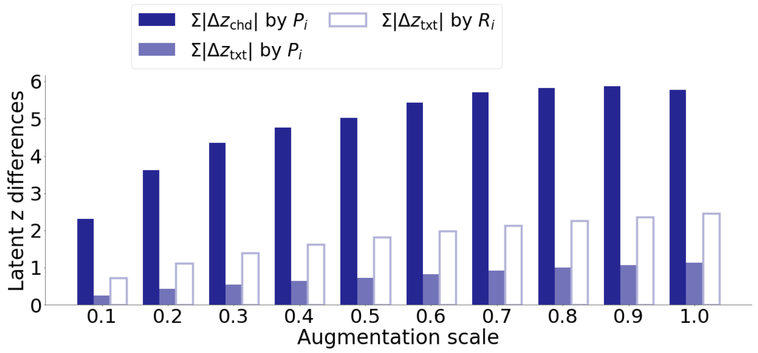
When z_{chd} and z_{txt} are well disentangled, small variations over the note pitches of the original music should lead to a larger change on z_{chd}, while variations of rhythm will have more influence on z_{txt}. Following this assumption, we adopt a *disentanglement evaluation via data augmentation* method used in [122] and further developed in [271].

We define F_i as the operation of transposing all the notes by i semitones, and use the L_1-norm to measure the change of latent z after augmentation. Figure 3.5(a) shows a comparison between $\Sigma|\Delta z_{\text{chd}}|$ and $\Sigma|\Delta z_{\text{txt}}|$ when we apply F_i to all the music pieces in the test set (where $i \in [1, 12]$).

It is conspicuous that when augmenting pitch in a small range, the change of z_{chd} is much



(a) A comparison between Δz_{chd}, Δz_{txt} after pitch transposition on all notes.



(b) A comparison among Δz_{chd}, Δz_{txt} after beat-wise pitch transposition and texture augmentation with different probabilities.

Figure 3.5: Results of objective measurement.

larger than the change of z_{txt}. At the same time, the change of z_{txt} gets higher as the augmentation scale increases. Similar to the result in [271], the change of z_{chd} reflects human pitch perception as z_{chd} is very sensitive to a tritone transposition, and least sensitive for a perfect octave.

We further define P_i as the function to randomly transpose all the notes in one beat either up or down one semitone under a certain probability i, and R_i as the function to randomly reduce the note duration by half. Figure 3.5(b) shows a comparison between $\Sigma|\Delta z_{\text{chd}}|$ and $\Sigma|\Delta z_{\text{txt}}|$ when we apply P_i and R_i to all the music pieces in our test set (where $i \in [0.1, 1.0]$).

For each value of i in the figure 3.5(b), the first and second bars demonstrate $\Sigma|\Delta z_{\text{chd}}|$ and $\Sigma|\Delta z_{\text{txt}}|$ caused by P_i function, while the third bar indicates $\Sigma|\Delta z_{\text{txt}}|$ caused by R_i function. (We did not show $\Sigma|\Delta z_{\text{chd}}|$ caused by R_i since they are all zero.) It again proves that the chord representation is more sensitive than texture representation under pitch variations, and conversely, texture representation is more sensitive than chord representation under rhythm variations.

3.4.3 Subjective Evaluation

Besides objective measurement, we conduct a survey to evaluate the musical quality of compositional style transfer (see Section 3.3.1). Each subject listens to ten 2-bar pieces with different chord progressions, each paired with 5 style-transfer versions generated by swapping the texture representation with a random sample from the test set. In other words, each subject evaluates 10 groups of samples, each of which contains 6 versions of textures (1 from the original piece and 5 from other pieces) under the same chord progression. Both the order of groups and the sample order within each group are randomized. After listening to each sample, the subjects rate them based on a 5-point scale from 1 (very low) to 5 (very high) according to three criteria: *creativity*, *naturalness*, and *musicality*.

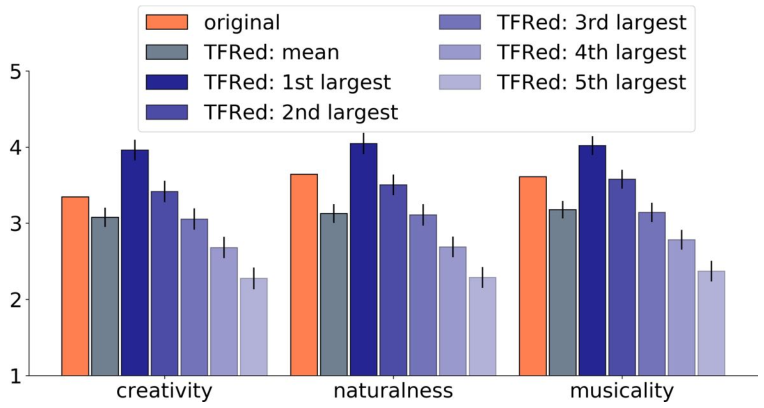


Figure 3.6: Subjective evaluation results. Here “TFRed: xth largest” denotes the x^{th} (largest) order statistic of the transferred segments.

A total of 36 subjects (26 females and 10 males) participated in the survey. Figure 3.6 shows the comparison result among the original pieces (indicated by the orange bars) and the transferred pieces in terms of their mean and *order* statistics. The heights of bars represent averaged ratings across the subjects, and the error bars represent the confidence intervals computed via paired t-test [102]. The result shows *if we randomly transfer a piece's texture 5 times, the best result is significantly better than the original version (with p-value < 0.005)*, and there are only marginal differences between the second-largest statistics and the original (with p-value > 0.05) in terms of creativity and musicality. We also see that, on average, the transferred results are still rated lower than the original ones. How to automatically decide the quality of a transferred result is considered future work.

3.5 Related Work

We review two techniques of automated music generation related to this chapter: controlled generation (Section 3.5.1) and representation disentanglement (Section 3.5.2). For a more general review of deep music generation, we refer readers to [26, 27].

3.5.1 Controlled Music Generation

Most existing learning-based methods regard controlled music generation as a *conditional estimation* problem. That is, to model $p(\text{music}|\text{control})$, in which both music and control are usually time-series features. Another approach closely related to conditional estimation is first to learn the joint distribution $p(\text{music}, \text{control})$ and later on *force* the value of control during the generation process.

The above two methods have been used in various tasks, including generating chords based on the melody [213], creating the melody based on the chords [37, 270], completing the counterparts or accompaniment based on the melody or chord [64, 65, 83, 108, 215, 289], and producing the audio waveform based on timbre features [107, 136].

However, many abstract music factors, such as texture and melody contour, could hardly be explicitly coded by labels. Even if such labels are provided, the control still does not allow continuous manipulation, such as sampling and interpolation. Consequently, it remains a challenging task to control music by more abstract factors without complex heuristics [139].

3.5.2 Music Representation Disentanglement

Learning disentangled representations is an ideal solution to the problem above, since: 1) representation learning embeds discrete music and control sequences into a continuous latent space, and 2) disentanglement techniques can further decompose the latent space into interpretable subparts that correspond to abstract music factors. Recent studies show that VAEs [125, 203] are in general an effective framework to learn the representations of discrete music sequences, and the key to a successful disentanglement is to incorporate proper inductive biases into the representation learning models [157].

Under a VAE framework, an inductive bias can be realized in various forms, including constraining the encoder [5, 162, 260], constraining the decoder [43], imposing multitask loss functions [30, 271], and enforcing transformation invariant results during the learning process [138, 171]. This study is based on our previous work [271] in which we disentangle pitch and rhythm factors for monophonic segments. We extend this idea to polyphonic composition, while the model design is more similar to [260].

3.6 Summary

In summary, we contributed an effective algorithm to disentangle polyphonic music representation into two interpretable factors, chord and texture, under a VAE framework. Such interpretable representations serve as an intuitive human-computer co-creation interface, by which we can precisely manipulate individual factors to control the flow of the generated music. In this paper, we demonstrated three ways to interact with the model, including compositional style transfer via swapping the latent codes, texture variation by sampling from the latent distribution, and accompaniment arrangement using downstream conditional prediction, among other possibilities. We hope this work can shed light on the field of controllable algorithmic composition in general, especially on the paradox between model complexity and model interpretability. We acknowledge that the learned music factors are still very basic. In the following chapters, we plan to extract more abstract and longer-range features using hierarchical models. We also plan to explore more ways to control the music generation for practical usage.

4 | Multimodal Disentanglement for Audio-to-Symbolic Music Generation

The previous chapters introduced methods for learning concept representations from symbolic music data. In this chapter, we explore how these concepts can be applied in a practical setting: can we automatically generate a piano accompaniment score from the audio of a pop song? This is the *audio-to-symbolic arrangement* task that we tackle in this chapter. If approached from a utility-based perspective, one might attempt to first transcribe the audio into a multi-track score and then apply piano reduction. However, this pipeline is highly challenging and usually lacks sufficient labels to guide the process effectively. From the perspective of concept manipulation, a more direct and interpretable approach is to transfer shared concepts, such as music texture, that are expressed differently across modalities, but correspond to the same underlying representation. This approach allows the arrangement to sound like the original audio, rather than strictly replicating it.

The multimodal model we propose builds upon the polyphonic disentanglement framework introduced in the previous chapter. It jointly encodes information about chord and texture from both audio and symbolic inputs. To enable audio-driven generation, we further introduce a tailored training strategy that gradually shifts the source of information from a corrupted symbolic score to the audio signal. By the end of training, the texture posterior conditioned on the score is reduced to a standard normal distribution, allowing the model to rely solely on audio during inference.

This chapter is based on the published work *Audio-to-Symbolic Arrangement via Cross-Modal Music Representation Learning* [245], conducted in collaboration with Dejing Xu, Gus Xia, and Ying Shan.[1]

4.1 Audio-to-Symbolic Arrangement

Piano arrangement is widely used in practice to reproduce various complex music signals. The key idea is to transform the original music, usually represented by an audio mixture or full score of a band, into a piano score (that can be performed using only two or four hands) without loss of major music information. For example, piano reductions are made for classical orchestral music, and piano covers are created for pop songs. A good arrangement is not merely a transcription of the original audio mapped onto the keyboard, but also a realization of the original content that makes musical sense as a composition in the new instrument.

In this chapter, our focus is on automatic *audio-to-symbolic* arrangement. Unlike automatic music transcription [16, 17], the task aims at the symbolic generation according to the audio content, which consists of an arbitrary set of instruments and may contain timbral effects that cannot be easily transcribed. Existing systems mainly rely on rule-based or simple statistical models [8, 195, 222, 223], where audio-analysis and symbolic-generation modules are more or less independent and loosely connected by some bottleneck audio features. Such design often suffers from two limitations. First, the arrangement patterns are usually very rigid since both the bottleneck audio features and the accompaniment patterns are largely pre-defined. Second, nuanced features such as groove patterns and bass lines are difficult to be modeled using existing

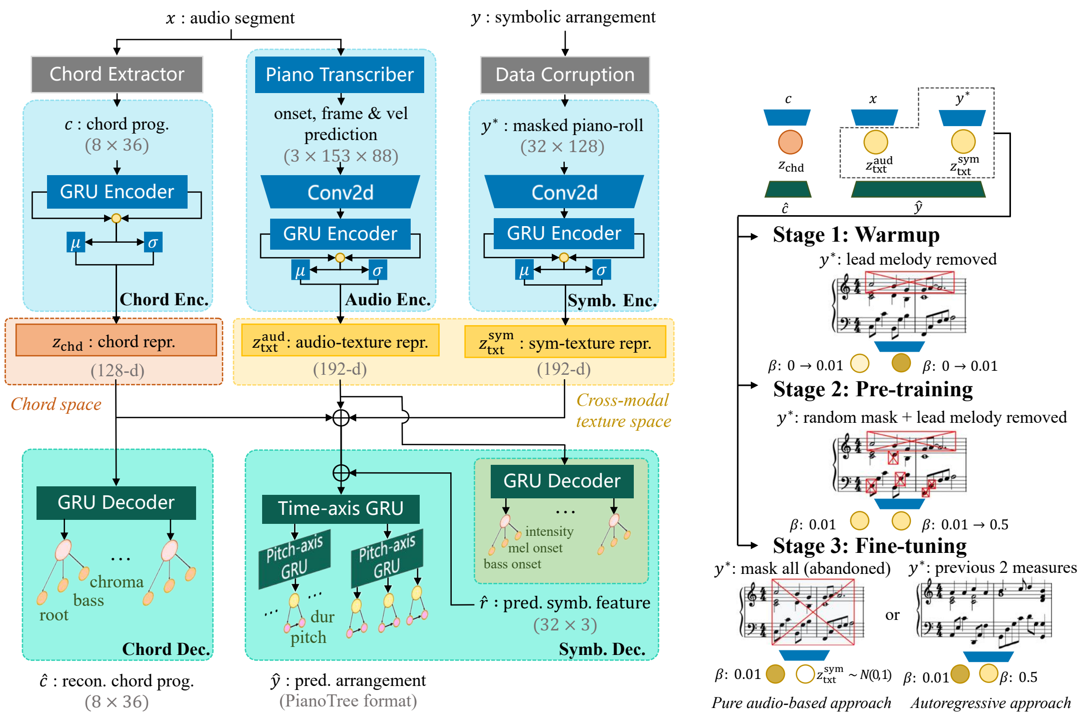
[1]Code and demos can be accessed via https://github.com/ZZWaang/audio2midi.

MIR techniques.

We consider end-to-end audio-to-symbolic deep generative modeling a better choice, as neural-based modeling potentially enables more flexible symbolic generation and an end-to-end architecture allows nuanced features to flow from audio to symbolic modal. On the other hand, we are faced with a great challenge—the relation between audio and its possible symbolic arrangement is essentially *one-to-many* and the prior knowledge of a good piano composition is only partially present in the audio. In other words, a naive supervised-learning model would easily get confused by the noisy audio-symbolic pairs and collapse to certain specific accompaniment patterns.

To solve the problem, we propose a cross-modal representation learning framework, in which the input is the audio of a pop-song accompaniment under arbitrary instrumentation, while the output is an arrangement in MIDI format. The model encodes a cross-modal representation from both audio and symbolic modals and decodes the information back to the symbolic domain. The latent representation contains the audio content and reflects prior knowledge about symbolic composition. During pre-training, we initialize the model to be an almost pure symbolic-to-symbolic variational autoencoder. By gradually corrupting the input and strengthening the variational constraints, the model is trained to lean more towards the audio. During the fine-tuning, we could provide the symbolic side with either Gaussian noise or information from previous bars to make the model fully dependent on audio or autoregressive.

In sum, we contribute the first end-to-end approach for automatic audio-to-symbolic arrangement. The quality of the generated samples is significantly higher than baselines and even rated higher than human compositions in terms of creativity. Moreover, the arrangement problem is tackled by a tailored training strategy that optimizes the supervised objective under an unsupervised cross-modal representation learning framework, which can be potentially generalized to other one-to-many supervised training tasks.


x : audio segment
y : symbolic arrangement
Chord Extractor
Piano Transcriber
Data Corruption
c : chord prog. (8 × 36)
onset, frame & vel prediction (3 × 153 × 88)
y^* : masked piano-roll (32 × 128)
GRU Encoder
Conv2d
μ
σ
Chord Enc.
Audio Enc.
Symb. Enc.
z_{chd} : chord repr. (128-d)
$z_{\text{txt}}^{\text{aud}}$: audio-texture repr. (192-d)
$z_{\text{txt}}^{\text{sym}}$: sym-texture repr. (192-d)
Chord space
Cross-modal texture space
GRU Decoder
root
bass
chroma
Chord Dec.
Time-axis GRU
Pitch-axis GRU
pitch
dur
intensity
mel onset
bass onset
$\hat{r}$: pred. symb. feature (32 × 3)
Symb. Dec.
$\hat{c}$: recon. chord prog. (8 × 36)
$\hat{y}$: pred. arrangement (PianoTree format)
c
x
y^*
z_{chd}
$z_{\text{txt}}^{\text{aud}}$
$z_{\text{txt}}^{\text{sym}}$
$\hat{c}$
$\hat{y}$
Stage 1: Warmup
y^*: lead melody removed
β: 0 → 0.01
β: 0 → 0.01
Stage 2: Pre-training
y^*: random mask + lead melody removed
β: 0.01
β: 0.01 → 0.5
Stage 3: Fine-tuning
y^*: mask all (abandoned)
y^*: previous 2 measures
or
β: 0.01
$z_{\text{txt}}^{\text{sym}} \sim N(0,1)$
β: 0.01
β: 0.5
Pure audio-based approach
Autoregressive approach

(a) Model architecture. (b) Training strategy.

Figure 4.1: The proposed model architecture and training strategy.

4.2 Method

We aim to learn *chord representation*, *audio-texture representation* from audio, and *symbolic-texture representation* from its paired symbolic arrangement. In this chapter, we consider 2-bar audio segments (provided with beat annotation) and symbolic arrangement in $\frac{4}{4}$ time signature. The symbolic arrangement is represented under $\frac{1}{4}$ beat resolution.

4.2.1 Model Architecture

Figure 4.1(a) shows the overall model architecture, which adopts an encoder-decoder architecture and contains five parts: 1) a chord encoder, 2) a chord decoder, 3) an audio encoder, 4) a symbolic encoder, and 5) a symbolic decoder. We consider our model a cross-modal extension of a symbolic-domain chord and texture disentanglement study [247].

The chord encoder adopts a GRU layer to encode a 128-d chord representation z_{chd} from a chord progression, which is extracted from the audio using an existing chord extraction algorithm [114]. The chord decoder, which reconstructs the input chord progression, is introduced as a mechanism to avoid *posterior collapse* of z_{chd}.

The input to the audio encoder is an 8-beat-long audio segment, time-stretched and resampled to 95 BPM with a sample rate of 16kHz. We first use a piano transcriber architecture [91] to embed the audio feature into a stack of piano-roll-like matrices of onset, frame, and velocity predictions. Then, we use a 2D convolution layer followed by a GRU layer to encode 192-d audio-texture representation $z_{\text{txt}}^{\text{aud}}$. The same model structure is applied to the symbolic encoder to extract a 192-d symbolic-texture representation $z_{\text{txt}}^{\text{sym}}$ from a corrupted ground truth piano-roll. The data corruption is controlled by a tailored training strategy introduced in Section 4.2.3.

The symbolic decoder takes in the concatenation of z_{chd}, $z_{\text{txt}}^{\text{aud}}$, and $z_{\text{txt}}^{\text{sym}}$ and decodes the symbolic arrangement in a hierarchical manner using the decoder module of PianoTree VAE [248], the state-of-the-art polyphonic representation learning model. Besides the latent codes, the PianoTree decoder also takes in a time series of symbolic features, which is predicted from $z_{\text{txt}}^{\text{aud}}$ *only* to enhance audio information retrieval. Specifically, we explicitly constrain $z_{\text{txt}}^{\text{aud}}$ to predict three symbolic features: *bass onset*, *melody onset*, and *rhythmic intensity*, which usually strongly correlate with audio rhythmic information of bass drum, lead melody, and groove patterns, respectively. Both bass onset and melody onset are time series of onset probabilities, and rhythmic intensity is a time series of scalar values. The predicted features are fed to the corresponding time steps of the time-axis GRU in the PianoTree decoder. A similar method is also used to achieve disentanglement in [271].

4.2.2 Training Objective

The loss terms in our model include 1) reconstruction losses of chord, arrangement, and symbolic features, and 2) KL losses between all three latent factors with standard normal distributions.

Our model is essentially a *conditional variational autoencoder*, since the loss function can be formalized as the *evidence lower bound* (ELBO) of the conditional probability $p(y|x)$, where x is the audio and y is the arrangement.

The posterior distribution of the conditional VAE is defined as the product of the three encoder models:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|x, y) := q_{\phi_1}(z_{\text{chd}}|c)q_{\phi_2}(z_{\text{txt}}^{\text{aud}}|x)q_{\phi_3}(z_{\text{txt}}^{\text{sym}}|y),$$

where $\mathbf{z} := [z_{\text{chd}}, z_{\text{txt}}^{\text{aud}}, z_{\text{txt}}^{\text{sym}}]$, and $\boldsymbol{\phi} := [\phi_1, \phi_2, \phi_3]$ denotes the encoder parameters. Note that the chord progression c is a deterministic transform from x and is therefore absent in $q_{\boldsymbol{\phi}}(\mathbf{z}|x, y)$. The reconstruction distribution is defined as the product of the three reconstruction terms:

$$p_{\boldsymbol{\theta}}(x|\mathbf{z}) := p_{\theta_1}(c|z_{\text{chd}})p_{\theta_2}(y|\mathbf{z}, r)p_{\theta_3}(r|z_{\text{txt}}^{\text{aud}}),$$

where $\boldsymbol{\theta} := [\theta_1, \theta_2, \theta_3]$ denotes the decoder parameters, r denotes the ground truth symbolic features, and $p_{\theta_1}(c|z_{\text{chd}})$ is interpreted as a regularizer to the output distribution. Finally, the loss function is:

$$\begin{aligned}\mathcal{L}_{\beta}(\boldsymbol{\theta}, \boldsymbol{\phi}; x) = &- \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|x,y)}\Big[\log p_{\boldsymbol{\theta}}(x|\mathbf{z})\Big] \\ &+ \beta\, \text{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|x, y)||p(\mathbf{z})),\end{aligned}$$

where $p(\mathbf{z})$ is a 512-d standard normal prior and β is the KL annealing parameter.

4.2.3 Training Strategy

We propose a training strategy (shown in Figure 4.1(b)) to balance information from the audio encoder and the symbolic encoder so that the training starts with the unsupervised symbolic reconstruction task and shifts to the supervised audio-to-symbolic task. Particularly, there are three stages:

Stage 1, Warm-up: The lead voice of the ground truth arrangement is masked, and β increases from 0 to 0.01 for all three latent factors. The model is therefore enforced to predict melody solely from the audio.

Stage 2, Pre-training: Besides melody, the rest of the notes are randomly masked under the probability ranging from 0.5 to 0.8, where lower pitches have a higher probability to be masked. Meanwhile, β increases from 0.01 to 0.5 for $z_{\text{txt}}^{\text{sym}}$ and keeps 0.01 for the other two factors. The model is expected to learn more information from the audio.

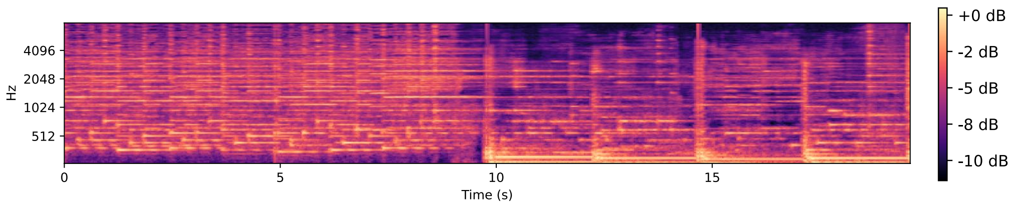
Stage 3, Fine-tuning: The model can be purely audio-dependent at this stage: we completely abandon the symbolic encoder by sampling $z_{\text{txt}}^{\text{sym}}$ from a standard normal distribution. Alternatively, we can also feed the arrangement of the previous two measures to the symbolic encoder to make the model autoregressive.

4.3 Experiments

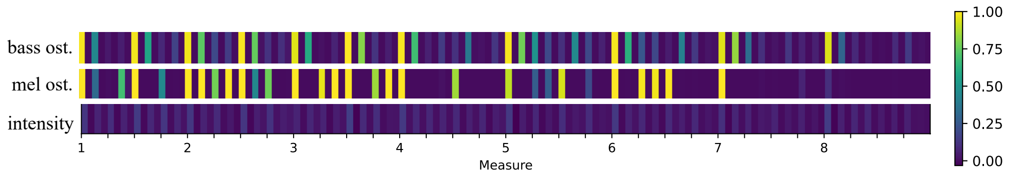
4.3.1 Implementation Detail

We train our model on the POP909 dataset [250], which contains about 1K MIDI files of pop song arrangements with time-aligned audios. We use the piano accompaniment MIDI tracks and keep the pieces with $\frac{2}{4}$ and $\frac{4}{4}$ meters and cut them into 8-beat music segments. The audio is also sliced into 8-beat segments, and the vocal is removed by the Spleeter source separation algorithm [95]. In all, we have 66K samples. We randomly split the dataset (at song level) into the training set (90%) and the test set (10%). All training samples are further augmented by transposing to all 12 keys. The chord, beat, and melody track annotations are all included in the dataset, while the ground truth bass onset is defined to be the occurrence of a MIDI pitch lower than 48, and rhythmic intensity is the number of simultaneous onsets normalized by a constant.

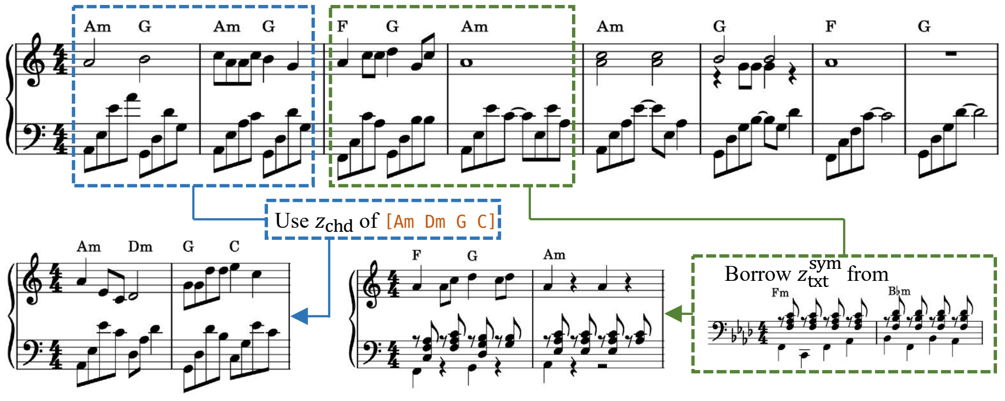
The piano transcriber used in our model is pre-trained on the MAESTRO dataset [90]. The model contains 62M trainable parameters in total, including 26M parameters in the piano transcriber. We use a batch size of 64 and Adam optimizer [124] with a scheduled learning rate from $4e-4$ to $4e-6$.



(a) Mel-spectrogram of an example input audio x.



(b) Predicted symbolic features of x.



Am
G
Am
G
F
G
Am
Am
G
F
G
Use z_{chd} of [Am Dm G C]
Am
Dm
G
C
F
G
Am
Borrow $z_{\text{txt}}^{\text{sym}}$ from
Fm
B♭m

(c) Arrangement result of x and examples of compositional style transfer.

Figure 4.2: An example of automatic arrangement based on the audio of an 8-bar excerpt from *1001 Nights* by Samuel Tai.

4.3.2 Arrangement Example

A 16-bar arrangement example (by predicting every 2 bars independently) is shown in Figure 4.2. The audio has a lead instrument and frequent bass note changes at the beginning, and a less intense groove halfway to the end (Figure 4.2(a)). These features are captured in the symbolic feature prediction (Figure 4.2(b)), as well as the symbolic arrangement (Figure 4.2(c), where we see melody with arpeggio texture in mm. 1–4, and arpeggio with decreasing intensity in mm. 5–8.

We also demonstrate that the model (after the pre-training stage) is capable of the *compositional style transfer* tasks [56] via replacement of the disentangled factors [247]. First, we change the chords in mm. 1-2 to another chord progression [Am, Dm, G, C] represented by z_{chd} (indicated by the blue arrow), and the generation changes to the desired progression while maintaining the original texture. Then, we replace mm. 3-4 with a new symbolic texture represented by $z_{\text{txt}}^{\text{sym}}$ (indicated by the green arrow), and the left-hand texture changes correspondingly while the harmony and the right-hand melody contour are kept unchanged.

4.3.3 Subjective Evaluation

We compare our proposed method with three baselines. The first two baselines adopt the common supervised approach, implemented with only the audio encoder and the PianoTree decoder with or without KL loss. The third baseline is solely chord-dependent, by setting $z_{\text{txt}}^{\text{aud}}$ to zero.

We invite people to subjectively rate the generation quality through a double-blind online survey. During the survey, the subjects listen to 6 groups of samples. In each group, the original audio is played, followed by the generated samples and the ground truth composition in random order. Both the order of groups and the sample order within each group are randomized. After listening to each sample, the subjects rate them based on a 5-point scale from 1 (very low) to 5 (very high) according to four criteria: *faithfulness* (to the original audio), *creativity*, *naturalness*,

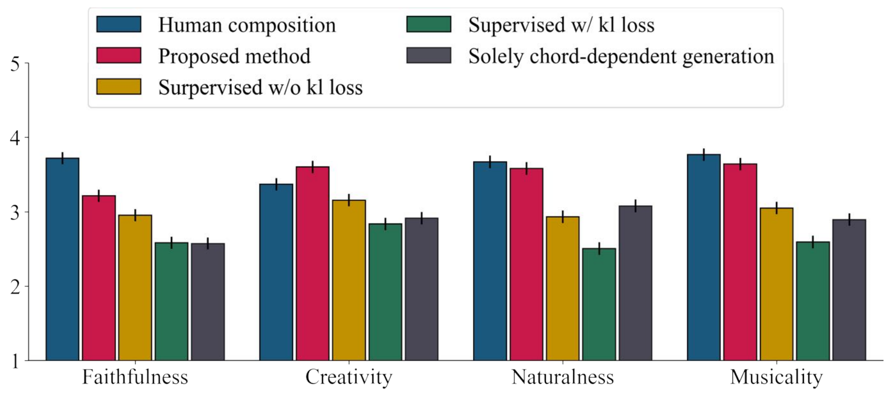


Figure 4.3: Subjective evaluation results.

and overall *musicality*.

A total of 26 subjects (8 females and 18 males) with different musical backgrounds have completed the survey. Figure 4.3 shows the result where the heights of the bars represent the means of the ratings and the error bars represent the confidence interval computed via within-subject ANOVA. The result shows that the proposed model is significantly better than the baseline models in terms of all four criteria, and the creativity is even significantly better than human composition (with p-value < 0.005).

4.4 Related Work

Automatic arrangement tasks can be conducted based on either symbolic or audio sources. Pure symbolic arrangement is commonly studied using deep generative models and has achieved considerable progress [27, 35, 215, 247]. In contrast, *audio-to-symbolic* arrangement, which is more related to this chapter, is still underresearched. Existing methods are mostly rule-based or rely on hand-crafted statistics. E.g., Takamori *et al.* extract chords and melodies from audio and pre-define several accompaniment textures [222, 223]. Song2Quartet [195] and Song2Guitar [8] further introduce matching probabilities between audio and score and use dynamic programming to search the notes. These models often lead to rigid patterns, and the musicality cannot yet serve for practical purposes.

An alternative shortcut to achieve arrangement is timbre style transfer [56, 68, 110, 150, 242], in which a latent timbre space is first learned and then transferred to piano timbre during inference. However, existing models are either constrained to monophonic instruments or based on synthetic audio data. Real-world audio is more complicated in instrumentation, and the integration of audio content with piano composition techniques is still an open problem.

4.5 Summary

We have contributed a cross-modal representation learning framework as the first end-to-end approach to accomplish the audio-to-symbolic automatic arrangement problem. Experimental results show that our model is able to capture harmonies, melody lines, and groove patterns from the audio without loss of musicality. The main novelty lies in the cross-modal training strategy that gradually shifts the input source from one modality to the other. We see this kind of tailored self-supervision control as a bridge between unsupervised learning tasks and supervised training. In the next chapter, we explore more flexible cross-modal methods using large language models to improve the quality and controllability of audio-to-symbolic conversion.

5 | Disentangling Style in Large Language Models

In previous chapters, we explored how musical concepts can be learned through disentangled representations, typically within a variational autoencoder framework. While these methods offer interpretable latent spaces corresponding to concepts, they often come at the cost of generation quality and flexibility. In contrast, large generative models have demonstrated superior fluency, structure, and coherence [3, 52, 160]. This raises a question: *can we incorporate disentangled concept representations into large generative models, thereby combining interpretability with high-quality generation?*

Among the concepts most relevant to music generation, style remains one of the most implicit and difficult to define. Though often described using text labels such as "swing," "classical," or "emotional," music style is not directly observable and hidden in concrete examples. In this chapter, we address this challenge by leveraging large pre-trained models to learn implicit styles from raw audio and apply the styles to symbolic music generation. Inspired by BLIP-2 [80], our model uses a Querying Transformer (Q-Former) to extract style representations from an audio LLM and further apply them to condition a symbolic LLM for generating piano arrangements. This approach can be viewed as a form of representation disentanglement in a broader sense, and we show that it achieves stronger results in audio-to-symbolic arrangement tasks compared to the method introduced in the previous chapter.

This chapter is based on original, unpublished work conducted in collaboration with Jingwei Zhao, Gus Xia, and Ye Wang.[1]

5.1 Representation Learning via Bootstrapping

Automatic music generation is often controlled by *explicit* content such as melody, chords, and text labels [18, 160, 247, 271, 286]. But music concepts can be more nuanced than we often realize. When musicians learn a style, instead of relying on abstract definitions like "romantic" or "jazz" alone, they absorb patterns from music examples that share common stylistic traits. The commonality across these examples forms a style—an *implicit* one that cannot be fully described with words or labels but only understood through the music itself. This chapter explores how this implicit style can be internalized from music examples and used to control music generation in a deep learning framework.

Large language models (LLMs) have shown strong capabilities in learning explicit music content, as demonstrated by probing studies [33, 165, 166, 240, 252] and adapter-based designs [151, 152, 259, 283]. Yet, control over implicit style remains limited. For example, when using audio to guide symbolic music generation, existing models can extract melody and chords [63, 245], but capturing stylistic traits like comping patterns or voicing preferences remains a greater challenge. This requires disentangling style from music content, which current LLM-based studies have yet to explore.

In this chapter, we explore learning implicit music style in a cross-modal setting for symbolic piano arrangement. Our goal is to generate an arrangement conditioned on two inputs: an audio example (providing style) and a lead sheet (melody and chords as content). To achieve this, we connect pre-trained LLMs in the audio and symbolic domains using a Querying Transformer (Q-Former), a lightweight Transformer originally designed for vision-language alignment [143].

[1]Demo page can be accessed via https://anonymous55aht.github.io/.

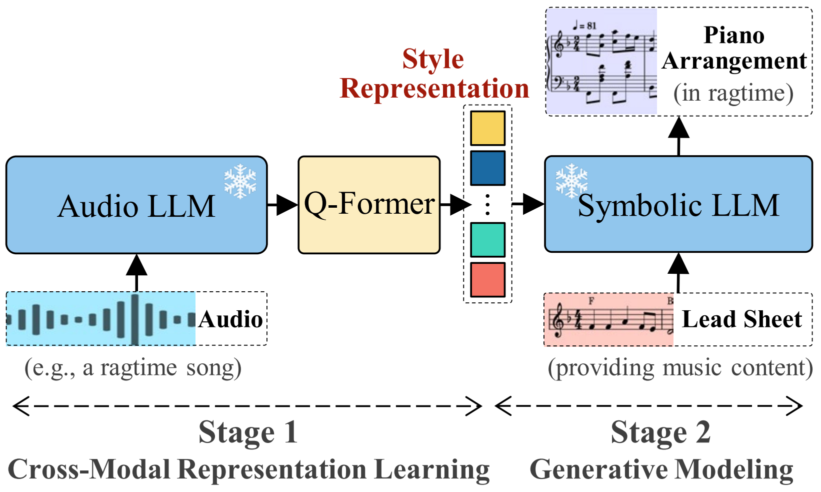


Figure 5.1: A Q-Former module bridges the modality gap between a frozen audio LLM and a symbolic music LLM. It extracts cross-modal music *style* from the hidden representations of the audio LLM and, together with a lead sheet providing music *content*, conditions the symbolic LLM for piano arrangement. The Q-Former is trained in a two-stage process, effectively bootstrapping audio-to-symbolic arrangement without re-training either LLM backbone.

As shown in Figure 5.1, we extend its role to capture abstract music style, extracting a style representation from the hidden states of the audio LLM. The symbolic LLM then conditions on this representation, along with the lead sheet, to generate an arrangement. The Q-Former enables style transfer between two unimodal LLMs without re-training them—a process we refer to as *bootstrapping*.

In our design, we treat the Q-Former as a bottleneck to transfer only style-related information and adopt a two-stage training strategy. The first stage employs contrastive learning, training the Q-Former to extract auditory representations that are musically relevant, expressible in piano composition, and independent of explicit music content. The second stage focuses on generative modeling, where the Q-Former's output conditions the symbolic LLM to arrange the desired piano performance. Experiments show that the complete system generates more stylistically accurate cover songs compared to existing audio-to-symbolic arrangement methods, besides achieving piano style transfer by specifying audio examples.

In sum, the contributions of this chapter are threefold:

1. We use the Q-Former to **align audio and symbolic modalities through abstract music style**, extending its role beyond content alignment in vision-language tasks.

2. We demonstrate **a new methodology to disentangle music style from pre-trained LLMs**, offering a more scalable alternative to traditional latent-variable disentanglement methods.

3. Our model achieves **style-preserving audio-to-symbolic piano cover arrangement and enables style transfer based on audio examples**. Experiments show that it outperforms existing audio-to-symbolic models, including both disentanglement-based methods and standard LM approaches.

5.2 Method

Our goal is to learn music style representations from the audio and leverage these representations to arrange symbolic piano performances. To bridge the modality gap from audio to symbolic music, we adopt the Q-Former [143] under a two-stage training strategy. Stage 1 focuses on audio-symbolic representation learning with a frozen audio LLM, while Stage 2 addresses audio-to-symbolic arrangement with a symbolic LLM. In Section 5.2.1, we first introduce our audio-symbolic data pairing method that facilitates style learning. We illustrate the Q-Former architecture in Section 5.2.2, followed by the details of the two-stage training procedure in Sections 5.2.3 and 5.2.4.

5.2.1 Data Pairing for Style Learning

Training an audio-to-symbolic alignment model requires paired audio–MIDI data. In this chapter, we use 10s audio clips paired with 4-bar MIDI segments. Our goal is not to model low-level note-to-note correspondence, but to extract style—and only style—from the audio modality.

Therefore, the audio and MIDI pairs are *loosely* aligned in the center of the segment but are not necessarily synchronized at the note level. This setup prevents the model from learning note transcription. To further encourage style abstraction, each MIDI segment is randomly transposed to all 12 keys during training.

We represent music audio as raw waveforms sampled at 32kHz. MIDI is tokenized into note event sequences quantized at 1/12-beat resolution. We include various symbolic features, including time signature (quadruple and triple meters), tempo curve, note pitch, duration, and velocity.

5.2.2 Q-Former Architecture

The Q-Former is a Transformer encoder architecture with two parallel, modality-specific streams that share the self-attention layers. As shown in Figure 5.2, it accepts both audio and symbolic inputs and aims to learn cross-model music style representation. The left stream interacts with the audio LLM to extract auditory music features. The right stream encodes symbolic music representations. A set of querying embeddings (randomly initialized) is fed to the left stream and serves as the bridge between the two modalities. These queries participate in three types of interactions: among themselves through self-attention, attending to the audio LLM via cross-attention, and attending to the symbolic stream to the right through the shared self-attention layers. To control the cross-modal interactions, we apply different self-attention masks according to specific training objectives, as detailed further in Section 5.2.3.

We initialize the Q-Former weights using the pre-trained MusicBERT-Base model [278]. The added cross-attention layers are randomly initialized. Following Blip-2 [143], we use 32 queries with dimension 768. Symbolic notes are embedded in the OctMIDI format [278], which learns a joint note-wise embedding by summing up the embeddings of individual note attributes. Overall, the Q-Former comprises 186M parameters, including the learnable queries and symbolic note embeddings.

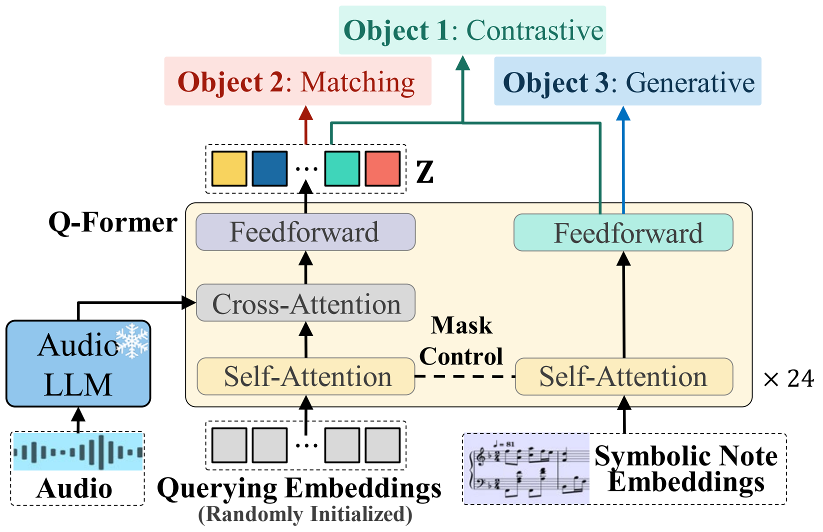


Figure 5.2: The Q-Former is a Transformer encoder with two parallel, modality-specific streams that share the self-attention layers. The left stream queries music style from an audio LLM via cross-attention, and the right stream encodes symbolic music. Cross-modal interactions are regulated by self-attention masks specific to training objectives.

5.2.3 Stage 1: Audio-Symbolic Representation Learning

In the representation learning stage, we leverage cross-attention to integrate Q-Former with MusicGen [52], one of the leading general music audio LLMs available today. The primary goal is for the queries to capture, from the audio, cross-modal representations that reflect music style. To bridge the modality gap between audio and symbolic music, we jointly train the Q-Former using three complementary objectives, each employing a distinct attention mask controlling the cross-modal interaction.

The primary objective is **Audio-Symbolic Contrastive Learning**, which enforces a higher audio-symbolic similarity for positive pairs compared to negative ones. Let $\mathbf{Z} \in \mathbb{R}^{32\times768}$ be the query outputs from the audio stream of Q-Former, and $t \in \mathbb{R}^{1\times768}$ be the output embedding of the start token (<s>) from the symbolic stream. We define the audio-symbolic similarity as $\max_k(\cos(\mathbf{Z}_k, t))$ for $k = 1, 2, \cdots, 32$, where $\cos(\cdot,\cdot)$ denotes the cosine similarity. The contrastive loss pulls closer aligned audio and symbolic clips in the representation space, while pushing apart unrelated pairs. To prevent information leakage, we employ a *unimodal self-attention mask*, ensuring queries and symbolic notes do not attend to each other. For details on mask configurations, we refer readers to BLIP-2 [143].

The second objective is **Audio-Symbolic Matching**. It is formulated as a binary classification task, where the model predicts whether a given audio-symbolic pair corresponds to each other. On top of contrastive loss, the matching loss aims to capture a finer cross-modal correspondence. In this case, we apply *no masking*, allowing the queries to attend across modalities. Each query output $\mathbf{Z}_k$ is fed into a binary linear classifier to produce a logit, and the logits from all queries are averaged to compute the final matching score. To create informative negative pairs, we employ the hard negative mining strategy in [142].

The final objective, **Audio-Grounded Symbolic Generation**, trains the Q-Former to autoregressively generate piano arrangement conditioned on an input audio and lead sheet. We implement a *cross-model causal self-attention mask*, allowing the symbolic notes to see the queries but not vice versa. This generative loss ensures that the auditory music style extracted by the queries can be translated to the symbolic modality. To signify a decoding task, we replace the starting <s> token with a <DEC> token prepended by a sequence of lead sheet note embeddings.

5.2.4 Stage 2: Audio-to-Symbolic Generative modeling

In the generative modeling stage, we take advantage of the generative capability of MuseCoco [160], a symbolic music LLM. As illustrated in Figure 5.3, MuseCoco is used to reconstruct a piano arrangement based on two concatenated conditional inputs: 1) the query output embeddings $\mathbf{Z}$ from the Q-Former, and 2) a lead sheet. The Q-Former is pre-trained at Stage 1 to extract cross-modal music style from the audio, thus providing style guidance. The lead sheet defines the theme melody and harmony as the content.

To enable compatibility with MuseCoco, we project $\mathbf{Z}$ into the same embedding dimension as MuseCoco's token embeddings using a linear layer. Symbolic note tokens are converted to the

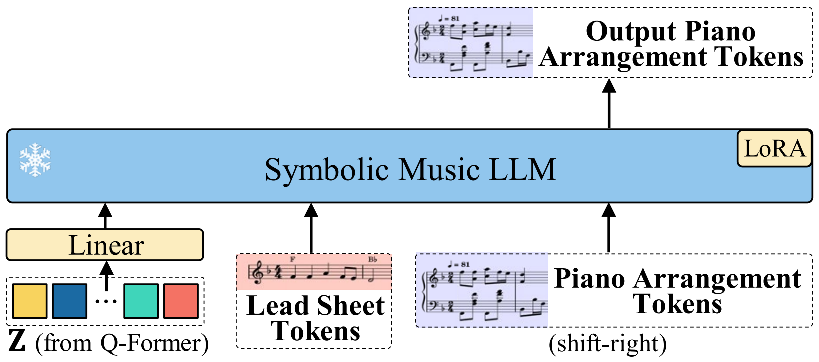


Figure 5.3: A symbolic music LLM is used to generate piano arrangement based on two inputs: the query output $\mathbf{Z}$ from the Q-Former with cross-modal music style, and a lead sheet providing music content. A LoRA adapter is incorporated to reweight self-attention and accommodate the newly introduced lead sheet condition.

REMI [108] format. Since MuseCoco does not natively support lead sheet conditioning and the inclusion of the lead sheet alters its input format, we incorporate a LoRA adapter [104] of rank 16 into each self-attention layer. This allows the model to reweight attention and accommodate the added conditioning inputs. Notably, MuseCoco itself remains frozen throughout this process.

5.3 Experiments

In this section, we evaluate the performance of our proposed audio-to-symbolic arrangement model. Section 5.3.1 describes the datasets used, and Section 5.3.2 details the model configurations. Our model takes as input a lead sheet and an audio reference. When the lead sheet and audio are paired, the task is piano cover generation; when they are unpaired, the task is style transfer. Section 5.3.3 presents qualitative demonstrations for both tasks. Given that prior work has primarily focused on piano cover generation, we benchmark our model on this task in Section 5.3.4. Finally, Section 5.3.5 presents an ablation study on the three pre-training objectives in the first stage of model training.

5.3.1 Datasets

Our model is trained on two dual-modal music datasets: POP909[249] and PIAST [10]. Specifically, POP909 contains 1K piano cover arrangements created by professional musicians. The music genre is primarily Chinese pop, while the accompanying audio features diverse band instrumentation, which can help the model learn generalizable audio representations of pop music. PIAST, on the other hand, contains 8K piano performance recordings along with symbolic transcriptions across a variety of genres, including pop, jazz, and classical. This diversity encourages the model to produce more expressive and stylistically varied performances. We split both datasets at the song level into training (90%), validation (5%), and test (5%) sets. Each symbolic MIDI file is clipped into 4-bar segments with a 2-bar hop size, transposed to all 12 keys, and center-aligned with the corresponding 10s audio clip.

We also test on two out-of-domain datasets: Ballroom [79, 132] and GTZAN [60, 238]. Both datasets feature audio recordings with diverse band and orchestral instrumentation, as well as fine-grained music genres such as jive and bossa nova. Since they lack paired symbolic annotations, we use them for inference only. This allows us to assess the model's generalization ability and its capacity to accommodate styles beyond pop music.

5.3.2 Model Configuration and Training Details

We use MusicGen-Large [52] as our audio LLM. We discard the text encoder and retain only the music decoder, a 48-layer Transformer. Audio codecs are fed to the decoder and we extract the hidden representations from the 25th layer, as prior probing studies [33, 165, 166, 240, 252] suggest that middle layers capture more musically meaningful features. This setup retains 1.7B frozen parameters from MusicGen.

For symbolic music arrangement, we adopt MuseCoco-xLarge [160], which is a 24-layer Transformer decoder pre-trained on large-scale symbolic music corpora. We remove its text-related components and keep 1.2B frozen parameters from the music decoder.

The Q-Former comprises 186M learnable parameters, which is significantly smaller than the billion-scale backbone models. In Stage 1, it is pre-trained in FP16 using a batch size of 128 for 10 epochs (130K iteration). The LoRA adaptor in Stage 2 adds 5M parameters, and we fine-tune the model for another 5 epochs using a batch size of 32. Both training stages are conducted on four RTX A40 GPUs (48GB each). We use the AdamW optimizer [158] with an initial learning rate of 1e-4, a linear warm-up over the first 1k steps, and a cosine decay schedule to a final rate of $1e-5$. During inference, we use top-k sampling with $k = 15$.

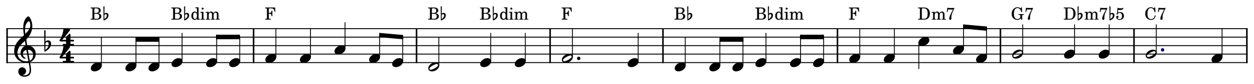
5.3.3 Arrangement Demonstration

In this section, we demonstrate the performance of our audio-to-symbolic arrangement model under *freely manipulated* audio style references. Figure 5.4(a) shows an 8-bar lead sheet excerpt from the musical *The Sound of Music*. The selected passage features harmonically rich chords, including diminished and seventh chord qualities, which present suitable complexity for arrangement experiments. Figures 5.4(b) to 5.4(d) showcase the arrangement results conditioned on varied audio references. The 8-bar arrangement is generated using windowed sampling, wherein a 4-bar context window progresses forward every 2 bars and continues sampling conditioned upon the preceding 2 bars.

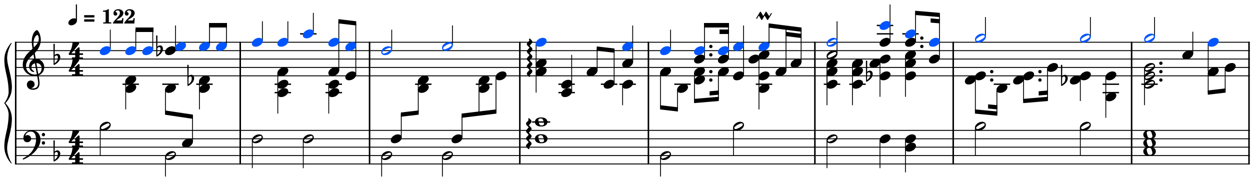
Figure 5.4(b) shows the piano cover from the original *The Sound of Music* soundtrack,[2] which features lush orchestration dominated by string ensembles. Our arrangement captures this orchestral essence through dense, block-chord voicing that emulates the sonority of string sections. Additionally, ornaments such as arpeggios and trills are found to complement the sweeping harmonic textures, which contribute to the free-flowing character of the music.

Figure 5.4(c) shows an arrangement conditioned on the ragtime classic *The Entertainer*.[3] Fol-

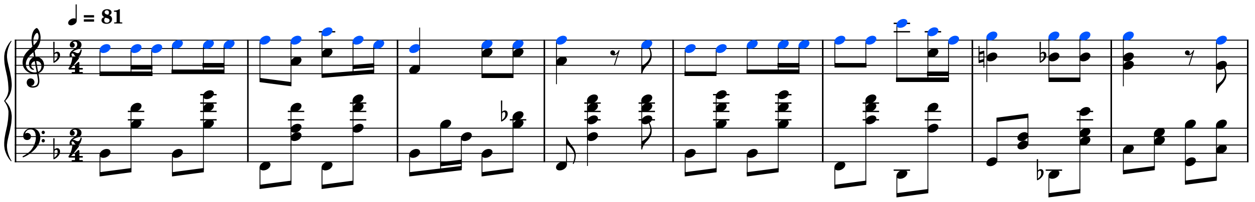
[2]Original audio: https://youtu.be/6f0T6UV-HiI&t=57
[3]Ragtime audio: https://youtu.be/jKlfNfRZL9I&t=11



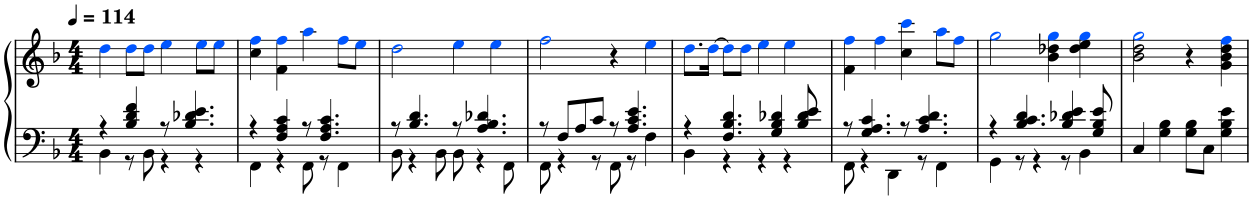
(a) The input lead sheet.



(b) Piano cover based on the original soundtrack.



(c) Piano arrangement in ragtime.



(d) Piano arrangement in bossa nova.

Figure 5.4: Audio-to-symbolic arrangement for an 8-bar excerpt from *The Sound of Music*. Figures 5.4(b) to 5.4(d) are arranged based on the lead sheet in 5.4(a) and an audio reference from the original soundtrack, a ragtime piece, and a bossa nova piece, respectively. Preserved music contents are highlighted in blue note heads. Synthesized audio is submitted in supplementary.

lowing the audio recording, the arrangement's tempo is "not fast," and the piano texture distinctly adopts a ragtime rhythm, featuring steady bass notes on downbeats and syncopated chordal accents on upbeats. Figure 5.4(d) shows an arrangement conditioned on the bossa nova piece *The Girl from Ipanema*.[4] In this interpretation, the arrangement is characterized by a moderate tempo and distinctive left-hand syncopated patterns characteristic of the bossa nova genre.

Across all three piano arrangements, while distinct music styles are effectively captured from the audio references, the theme melody and harmonic structures remain faithfully preserved. In Figure 5.4, we highlight melody notes preserved from the lead sheet using blue note heads.

[4]Bossa nova audio: https://youtu.be/DvA_wDOVD10&t=12

5.3.4 Evaluation on Piano Cover Generation

We evaluate our model both objectively and subjectively on the piano cover generation task, where the input is an audio clip and the output is a symbolic piano performance. To ensure a fair comparison with baseline models, we use Sheetsage [62, 63] to transcribe lead sheets from the audio, making audio the sole input for all methods. We consider two evaluation settings: 1) *in-distribution* evaluation on 46 test samples from the POP909 dataset, and 2) *out-of-distribution* evaluation on 96 tracks from the Ballroom and GTZAN datasets, reflecting the model's generalization to unseen genres and instrumentation. We use 16-bar audio excerpts for evaluation on POP909, and full 30s audio clips for Ballroom and GTZAN.

Baseline Models

We compare our model against two representative piano cover generation models: *PiCoGen2* [225] and *Audio-to-MIDI* [245], as well as one ablation variant of our method.

PiCoGen2 (PCG2) is a Transformer-based language model that builds on the hidden representations of Sheetsage, which itself is derived from Jukebox [58], a large-scale music language model. Leveraging Jukebox's internalized understanding of music *content*, PiCoGen2 generates symbolic piano arrangements directly from audio.

Audio2MIDI (A2M) is a disentanglement framework, using separate modules to extract piano texture and chord from the audio. The texture extractor is initialized from a pre-trained piano transcription model [91]. The extracted components are then merged to form a piano arrangement.

Ours w/o Pre-Training (w/o PT) is an ablation variant of our model in which the Q-Former is trained directly in Stage 2, without undergoing the representation learning phase in Stage 1. This setup tests the validity of the two-stage training strategy we applied in this work.

Objective Evaluation

We evaluate our model's performance in terms of *audio-to-symbolic coherence*, specifically assessing how well the generated piano covers preserve the *feel* of the original audio recordings. To do this, we repurpose the pre-trained Q-Former in Section 5.2.3 as a cross-modal retrieval model: for each audio clip in the test set, the Q-Former computes the similarity between the audio and all generated symbolic pieces. If the closest (i.e., most similar) symbolic output corresponds to the one generated from that audio input, we count it as a correct match.

Based on this setup, we report two metrics: 1) *Retrieval Accuracy*: the proportion of audio references for which the corresponding symbolic output is ranked closest. Higher values indicate stronger coherence; 2) *Mean Rank*: the average rank position of the correct audio-symbolic pair in all candidates. Lower values indicate better alignment.

	POP909		**Ballroom/GTZAN**	
	Accuracy $\uparrow$	**Rank** $\downarrow$	**Accuracy** $\uparrow$	**Rank** $\downarrow$
Ours	0.31 ± 0.02	6.57 ± 0.34	$\mathbf{0.22} \pm 0.01$	$\mathbf{13.56} \pm 0.54$
w/o PT	0.14 ± 0.01	8.91 ± 0.29	0.14 ± 0.01	21.59 ± 0.58
PCG2	0.21 ± 0.01	9.15 ± 0.36	0.06 ± 0.01	27.25 ± 0.32
A2M	$\mathbf{0.33} \pm 0.01$	$\mathbf{5.24} \pm 0.28$	0.10 ± 0.01	21.80 ± 0.23

Table 5.1: Objective evaluation on audio-to-symbolic coherence. Ballroom and GTZAN serve as out-of-distribution datasets to assess cross-domain arrangement capabilities.

We conduct experiments separately on POP909 test set and Ballroom/GTZAN dataset. For each test piece, we randomly select a 4-bar segment from the full-length generation and the corresponding 10s audio clip. This process is repeated over 10 independent sampling rounds and we report the mean and standard error. As shown in Table 5.1, while our model performs comparably to Audio2MIDI on the in-distribution POP909 dataset, ours significantly outperforms all baselines on the out-of-distribution Ballroom and GTZAN datasets. This highlights our model's capability to generalize across styles and genres, enabling effective style transfer and cross-domain piano arrangement.

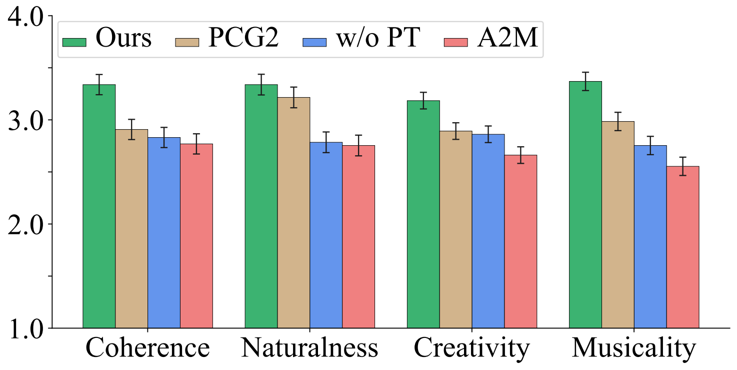


Figure 5.5: Subjective evaluation on music quality, conducted on Ballroom and GTZAN datasets to assess the musicality for diverse music genres and styles.

Subjective Evaluation

We conduct a double-blind online listening survey to evaluate the music quality. The survey comprises 6 test pieces of varied genres drawn from the Ballroom and GTZAN datasets. Each test piece is accompanied by 4 piano covers interpreted by different models. For each model, we select the best result from 3 generated samples. All samples are 16 bars long and rendered to audio using the default Cakewalk soundfont, resulting in ~40s audio per sample. Both the order of the test pieces and the order of the samples are randomized. Participants are asked to complete 3 test pieces by rating each piano cover on a 5-point Likert scale across 4 criteria: 1) *Audio-to-Symbolic Coherence*, 2) *Naturalness*, 3) *Creativity*, and 4) *Overall Musicality*.

A total of 21 participants with diverse music backgrounds have completed our survey, with an average completion time of 12 minutes. The mean ratings and standard errors, computed by within-subject ANOVA [208], are presented in Figure 5.5. Significant differences are observed across all criteria (p-value $p < 0.05$). While our model performs comparably to the state-of-the-art PiCoGen2 in terms of *Naturalness*, it consistently outperforms all baselines across criteria by a clear margin. These results align with the objective evaluation and suggest that our model more effectively captures music style and produces coherent, high-quality piano arrangements.

5.3.5 Ablation Study on Pre-Training Objectives

To evaluate the contribution of each pre-training objective to cross-modal representation learning, we conduct an ablation study on the Q-Former's audio-to-symbolic retrieval performance after Stage-1 training. We test three configurations: contrastive loss only (**C**), contrastive + matching losses (**C+M**), and contrastive + matching + generative losses (**C+M+G**). We compute *Retrieval Accuracy* and *Mean Rank* using the Q-Former on 128 randomly sampled audio-symbolic 4-bar pairs from the test sets. Each experiment is repeated over 10 independent rounds, and we report the mean and standard error.

	PIAST		**POP909**	
	Accuracy $\uparrow$	**Rank** $\downarrow$	**Accuracy** $\uparrow$	**Rank** $\downarrow$
C	0.95 ± 0.00	2.60 ± 0.33	0.35 ± 0.01	5.18 ± 0.31
C+M	0.96 ± 0.00	$\mathbf{2.19} \pm 0.24$	0.42 ± 0.01	4.94 ± 0.39
C+M+G	$\mathbf{0.96} \pm 0.00$	2.26 ± 0.26	$\mathbf{0.45} \pm 0.01$	$\mathbf{4.64} \pm 0.37$

Table 5.2: Ablation study on the impact of individual pre-training objectives to cross-modal representation learning. **C**, **M**, and **G** denote the use of the Audio-to-Symbolic Contrastive, Matching, and Generative losses, respectively.

Evaluation is conducted separately on PIAST and POP909. The former involves piano-only music, while the latter includes multi-instrumental accompaniments, requiring the model to extract style from richer audio textures. As shown in Table 5.2, the performance difference is relatively small on PIAST, suggesting that contrastive learning alone may suffice for simpler piano alignment. However, on POP909, we observe that both the matching and generative losses contribute meaningfully to improved retrieval accuracy and lower mean rank. These findings indicate that all three objectives are important for learning robust, generalizable cross-modal representations.

5.4 Related Work

We review two areas of key relevance to our work. Section 5.4.1 discusses recent advances in LLMs for music generation. Section 5.4.2 focuses on piano cover generation, which is the primary task addressed in this chapter.

5.4.1 Music LLMs

Rapid progress in large language models (LLMs) has transformed how we interact with various forms of media, including text, image, and music [6, 130, 143, 235, 276]. In particular, music LLMs [3, 18, 173, 230] have notably influenced creative practices and user experiences. Models like MusicGen [52] can generate music audio with rich timbres directly from text, while MuseCoco [160] produces symbolic compositions with well-structured textures in varied genres. These advancements are driven by training large-scale neural networks on extensive data, scaling up to billions of model parameters to enhance controllability and musicality.

Despite these successes, most existing music LLMs operate in an unimodal setting, focusing solely on either audio or symbolic representations. Although text-to-music generation has become increasingly effective [3, 18, 52, 160, 173], text descriptions often fall short in expressing nuanced music styles and performance subtlety. In contrast, our work explores a cross-modal framework that bridges audio and symbolic modalities. By leveraging the strong perceptual understanding of audio LLMs and the expressive composition capabilities of symbolic LLMs, we bootstrap a system for audio-to-symbolic arrangement. This approach enables more intuitive and fine-grained control over music style beyond what can be conveyed through text alone.

5.4.2 Piano Cover Generation

Piano cover generation aims to reinterpret an audio recording as a symbolic piano performance. Unlike traditional music transcription, which primarily analyzes note-level content such as pitch and timing [76, 82, 129, 187, 279], a piano cover often targets higher-level, more structured music elements that shape the *feel* of a performance. The goal is to generate symbolic arrangements that not only sound correct but also feel musically aligned with the original audio.

Existing approaches to piano cover generation often leverage pre-trained transcription models, primarily extracting melodic and harmonic content from the audio [42, 178, 224, 225, 245]. However, such models tend to overlook stylistic nuances, resulting in outputs accurate in harmony but lacking the expressive character of the source performance. In this chapter, we re-frame piano cover generation through the lens of content-style disentanglement, acquiring *content* in the symbolic form (i.e., melody and chord progression) while learning *style* from the audio. This approach bridges the audio-symbolic gap more effectively, capturing not just *what* is played, but *how* it is played.

5.5 Summary

In this chapter, we introduce a cross-modal framework for audio-to-symbolic arrangement that extracts and applies implicit music style in pre-trained music LLMs. By repurposing the Q-Former to align audio and symbolic modalities through abstract style, our model enables expressive piano arrangement conditioned on both a lead sheet and an audio reference. Through a bootstrapping process, we extract stylistic features from an audio LLM and guide a symbolic LLM without re-training either backbone. Our experiments on piano cover generation and style transfer demonstrate improved audio-to-symbolic quality, highlighting the potential of this framework for controllable, style-aware music generation beyond explicitly labeled content.

Part II

Concept Organization

6 | Compositional Hierarchy

In Part I, we focus on learning specific representations corresponding to individual concepts of interest. Revisiting Figure 1.1, this corresponds to learning the individual concepts inside the broader ontology of music. However, music involves an abundance of concepts, which cannot be exhaustively enumerated. Rather than learning every concept in isolation, we must first address how to organize them.

This leads us to the second concept alignment problem explored in this thesis: *concept organization*. As illustrated in Figure 1.1, there are two complementary types of organizations. The first is *compositional hierarchy*, which captures how musical content is abstracted and structured across time. The second is *concept hierarchy*, which reflects how high-level ideas, such as mood or style, are gradually transformed into concrete musical material.

Chapter 6 focuses on building compositional hierarchy, while Chapter 7 addresses concept hierarchy. These chapters investigate whether explicitly modeling these hierarchies with music domain knowledge can produce longer-term, better-structured music with greater computational efficiency.

In this chapter, we explore compositional hierarchy through the task of pop music generation with a vocal melody and a piano accompaniment. We define a hierarchical language of pop songs, in which each level of hierarchy focuses on the semantics and context dependency at a certain music scope. The high-level languages reveal whole-song form, phrase, and cadence, whereas the low-level languages focus on notes, chords, and their local patterns. A cascaded diffusion model is trained to model the hierarchical language, where each level is conditioned on its upper levels.

This chapter is based on the published work *Whole-Song Hierarchical Generation of Symbolic Music Using Cascaded Diffusion Models* [243], conducted in collaboration with Lejun Min and Gus Xia.[1] This chapter builds on a melody reduction algorithm, which is discussed in Appendix B. Additional experimental details and extended discussions are provided in Appendix C.

6.1 Compositional Hierarchy in Music Composition

In recent years, we have witnessed a lot of progress in the field of deep music generation. With significant improvements on the quality of generated music [52, 230] on short segments (typically ranging from a measure up to a phrase), researchers start to put more emphasis on *long-term structure* as well as how to *control* the generation process in a musical way. The current mainstream approach of structural generation involves first learning disentangled latent representations and then constructing a predictive model that can be controlled by the learned representations or external labels [36, 247, 254, 271]. However, generating an entire song remains an unresolved challenge. As compositions extend in length, the number of involved music representations and their combinations grow exponentially, and therefore, it is crucial to organize various music representations in a structured way.

We argue that *compositional hierarchy* of music is the key to the solution. In this study, we focus on symbolic pop songs, proposing a computational *hierarchical music language* and modeling such language with cascaded diffusion models. The proposed music language has four levels. The top-level language describes the phrase structure and key progression of the piece. The second-level language reveals music development using a reduction of the melody and a rough chord progression, focusing on the music flow within phrases. The third-level language consists of the complete lead melody and the finalized chord progression, which is usually known as a

[1]We release the complete source code and model checkpoints at https://github.com/ZZWaang/whole-song-gen. The demo page is available at https://wholesonggen.github.io.

lead sheet, further detailing the local music flow. At the last level, the language is defined as piano accompaniment. Intuitively, the language aims to characterize the intrinsic homophonic and tonal features of most pop songs— a verse-chorus form, a chord-driven tonal music flow, and a homophonic accompaniment texture.

We represent all levels of the symbolic languages as multi-channel images and train four layers of image diffusion models in a cascaded fashion, one for each level of the music language. The generation scope of the first layer is full song and up to 256 measures, the scope of the second layer is 32 measures, and the third and fourth layers each have a scope of 8 measures. Additional autoregressive controls are added to the low-level diffusion models to strengthen long-term temporal coherence. Experimental results show that our model is capable of generating well-structured full-piece music with recognizable verse-chorus structure and high music quality.

Moreover, at each level, optional *external conditions* can be added via the cross-attention mechanism of diffusion models to control the generation process at each level of the hierarchy. As a demonstration, we add long-term control of chord progression, local control of rhythmic and accompaniment pattern to the corresponding levels of the hierarchy. All the external controls use pre-trained latent codes from existing music representation learning models. We show that these controls can effectively guide hierarchical generation in a more customizable way.

In summary, the contribution of the chapter is as follows:

1. **We achieve high-quality and well-structured whole-song generation** with cascaded diffusion models. Objective and subjective measurements show that both monophonic lead sheets and polyphonic accompaniment generated by our model have more identifiable phrase boundaries, better-structured phrase development in similarity and contrast, and higher music quality compared to baselines.

2. **We propose a computational hierarchical music language** as a structural inductive bias, making the training process decomposable and efficient in terms of data and com-

puting power utilization. Also, the hierarchical languages can be extracted automatically without manual annotation of music structure.

3. **Our model enables flexible and interpretable controls**, with not only our proposed hierarchical language but also with external pre-trained latent representations, such as chord, melodic rhythm, and accompaniment texture.

6.2 Methodology

Our model for whole-song generation is a realization of the music compositional hierarchy. In this section, we first introduce the definition of our hierarchical music languages in Section 6.2.1. Then, we discuss how to model these languages via cascaded diffusion models, where each level of the language is conditioned on its upper levels. We show the data representation of these languages in Section 6.2.2. The training and inference of the model are discussed in Section 6.2.3 and Section 6.2.4, respectively.

6.2.1 Definition of Hierarchical Music Languages

We define a hierarchical music language with four levels to reveal the generative procedure of music, as shown in Table 6.1. The highest level, *Form*, includes music keys and phrases. This is followed by *Reduced Lead Sheet*, which contains reduced melody and simplified chords. The third level, *Lead Sheet*, includes the lead melody and chords. The final level, *Accompaniment*, consists of the piano accompaniment. The key idea behind this hierarchical design lies in the relationship among the four levels—more abstract music concepts at higher levels are realized by stylistic specifications at lower levels. For example, a lead sheet is an abstraction implying many possible ways to arrange the accompaniment that share the same melodic and harmonic structure, while an instantiated accompaniment is one of the possible realizations showing the accompaniment structure in more detail.

Languages (res.)	**Specification**	**Data Representation**	**Structural Focus**
Form (m)	Key changes Phrases	$\boldsymbol{X}^1 \in \mathbb{R}^{8\times M\times 12}$	Music form
Reduced Lead Sheet (b)	Melody reduction Simplified chord	$\boldsymbol{X}^2 \in \mathbb{R}^{2\times \gamma M\times 128}$	Phrase similarity, phrase development & cadence
Lead Sheet (s)	Lead melody Chord	$\boldsymbol{X}^3 \in \mathbb{R}^{2\times \delta\gamma M\times 128}$	Melodic pattern, similarity & coherence
Accompaniment (s)	Accompaniment	$\boldsymbol{X}^4 \in \mathbb{R}^{2\times \delta\gamma M\times 128}$	Acc. pattern, similarity & coherence, Mel-acc relations

Table 6.1: Definition of the four-level hierarchical music language. We use m for measure, b for beat, s for step, to represent the temporal resolution. M denotes the number of measures in a piece, γ denote the number of beats in a measure, and δ denotes the number of steps in a beat.

Note that for Form, Lead Sheet, and Accompaniment, there are established music information retrieval algorithms for labeling. In contrast, the Reduced Lead Sheet is a unique design, which we refer the readers to Appendix C.1 for details.

6.2.2 Data Representation

While music scores are inherently symbolic, we transform them into continuous, image-like piano-roll representations for better compatibility with diffusion models. Specifically, languages at all levels are represented by multi-channel images (examples are shown in Appendix C.1). The image width represents sequence length under different resolutions, and the height represents 128 MIDI pitches or 12 pitch classes. We denote the piece length to be M measures, each measure containing γ beats, and each beat containing δ steps. In this chapter, we consider $\gamma \in \{3, 4\}$ and $\delta = 4$.

The language Form is a sequence of keys and phrases under the resolution of one measure. Keys are represented by $\boldsymbol{K} \in \mathbb{R}^{2\times M\times 12}$, where tonic information and scale information are stored on the two channels with binary values. For phrases, we use $\boldsymbol{P} \in \mathbb{R}^{6\times M\times 1}$, where six channels correspond to six phrase types (e.g., verse and chorus, see Table C.1 for more detail), and the pixel values indicate measure countdown. Formally, let $m_0, ..., m_0 + L - 1$ be the indices of a

L-measure phrase of type i_0, then for $m_0 \leq m < m_0 + L$,

$$\boldsymbol{P}[i, m, :] := \mathbb{1}_{\{i=i_0\}}(1 - \frac{m - m_0}{L}). \tag{6.1}$$

We broadcast $\boldsymbol{P}$ to match the pitch-axis of $\boldsymbol{K}$ and define the first-level language Form as $\boldsymbol{X}^1 := \mathrm{concat}(\boldsymbol{K}, \boldsymbol{P}) \in \mathbb{R}^{8 \times M \times 12}$. The other levels of languages use a piano-roll representation. Reduced Lead Sheet is represented by $\boldsymbol{X}^2 \in \mathbb{R}^{2 \times \gamma M \times 128}$ under the resolution of one beat, where two channels correspond to note onset and sustain. Both melody reduction and simplified chord progression share the same piano-roll using different pitch registers. Similarly, Lead Sheet uses $\boldsymbol{X}^3 \in \mathbb{R}^{2 \times \delta\gamma M \times 128}$ to represent the actual melody and chords, and Accompaniment uses $\boldsymbol{X}^4 \in \mathbb{R}^{2 \times \delta\gamma M \times 128}$ to represent the accompaniment, both in the same resolution of one step.

Note that for the four levels $k = 1, \ldots, 4$, $\boldsymbol{X}^k$ have different shapes. In the following sections, we write $\{\boldsymbol{X}^k | k \subset \{1, 2, 3, 4\}\}$ to denote the concatenation along the channel axes with possible broadcasting and repetition operations. For example, $\boldsymbol{X}^1$ can be expanded γ times in width and repeated 11 times in height to be concatenated with $\boldsymbol{X}^2$, resulting in a tensor $\boldsymbol{X}^{\leq 2} \in \mathbb{R}^{10 \times \gamma M \times 128}$. Additionally, we write the time-series expression $\boldsymbol{X}_t^k$ to denote $\boldsymbol{X}^k[:, t, :]$ for simplicity.

6.2.3 Model Architecture

Whole-song music generation is achieved by generating the four levels of hierarchical music languages one after another in a top-down order (as shown in Figure 6.1). For each level, we train a diffusion model to realize the current-level language based on the existing upper-level languages. The *actual scopes* (image widths) of these diffusion models are generally the same, yet the *music scopes* vary significantly since the resolution in lower-level languages is finer. In this chapter, for level $k = 1, ..., 4$, we set the actual scope b_k to be $b_1 = 256$ and $b_{2:4} = 128$, which means the music scopes for these levels are 256 measures, 128 beats, 128 steps, and 128 steps, respectively. In the usual setting when $\gamma = \delta = 4$, the music scopes of the models are 256

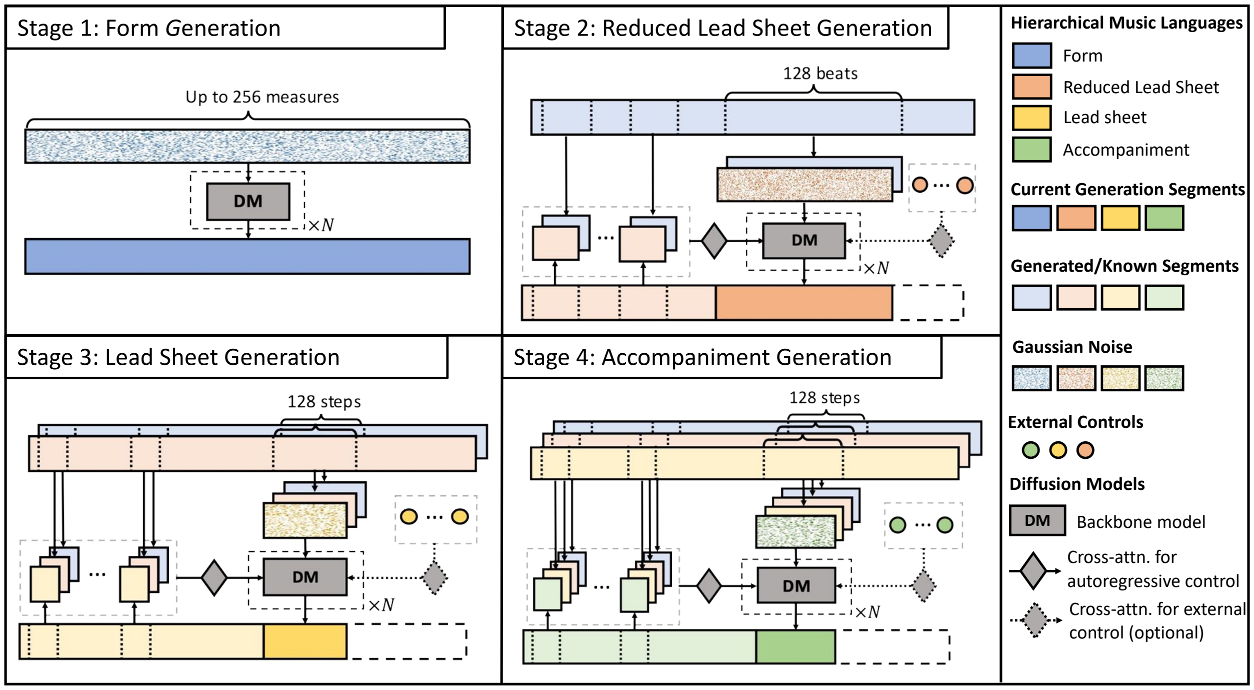

Stage 1: Form Generation
Up to 256 measures
DM
×N
Stage 2: Reduced Lead Sheet Generation
128 beats
Stage 3: Lead Sheet Generation
128 steps
Stage 4: Accompaniment Generation
Hierarchical Music Languages
Form
Reduced Lead Sheet
Lead sheet
Accompaniment
Current Generation Segments
Generated/Known Segments
Gaussian Noise
External Controls
Diffusion Models
Backbone model
Cross-attn. for autoregressive control
Cross-attn. for external control (optional)

Figure 6.1: The diagram of cascaded diffusion models for hierarchical symbolic music generation.

measures, 32 measures, 8 measures, and 8 measures, respectively. Consequently, except that the first layer is an unconditional generation of the whole sequence, the generation at all the other layers is essentially conditional generation of music segments sliced from the entire sequences.

The generation of a music language slice $\boldsymbol{X}^{k}_{t:t+b_k}$ at level $k \neq 1$ can be conditioned on multiple resources inside and outside the defined hierarchy. In this study, our model is designed to take in three sources of structural conditions:

Background condition. We regard the generation as a realization of existing higher-level languages at the corresponding scope $\boldsymbol{X}^{<k}_{t:t+b_k}$, where the higher-level language segments are like sketch images directly guiding the current generation. Background condition is applied by concatenating the input with $\boldsymbol{X}^{<k}_{t:t+b_k}$ along the channel axis.

Autoregressive condition. The segment should not only be a realization of the background condition, but also coherent with prior realizations $\boldsymbol{X}^{\leq k}_{<t}$. For example, the realization of a verse phrase at the end of the composition is usually similar to the realization in the beginning. In

our model, we make an autoregressive assumption that $\boldsymbol{X}^k_{<t}$ are known. We select S_k relevant music segments prior to t based on a defined similarity metric on $\boldsymbol{X}^1$. These music segments are encoded into latent representations and are cross-attended in the diffusion models. See Appendix C.2 for more details about the selection process.

External condition. Besides the compositional hierarchy, music generation can be controlled by other external conditions. These conditions can be stylistic controls of multiple scopes [247, 253, 271], or cross-modality controls of text [282] or audio [245]. As an illustration of our model compatibility, we use pre-trained latent representations of chords, rhythmic pattern, and accompaniment texture as the control for Reduced Lead Sheet, Lead Sheet, and Accompaniment generation, respectively. At each level k, we denote the array of external latent codes by $\boldsymbol{Z}^k_{t:t+b_k}$, which are cross-attended in our diffusion models.

For all four levels, we adopt a 2D-UNet with cross-attention similar to [174] as the backbone neural architecture and make several modifications. First, the input channels are increased to allow background condition. Second, autoregressive and external conditions are fed through cross-attention layers with classifier-free guidance [14]. In Appendix C.2, we include more details on the model architecture and training. Mathematically, we use diffusion to model the conditional probability of multiple levels of music segments. Let the backbone model at level k be denoted by

$$\epsilon_{\theta_k}(x_n, n, y^{\text{bg}}, y^{\text{ar}}, y^{\text{ext}}), \tag{6.2}$$

where θ_k is the model parameter, $n = 0, ..., N$ is the diffusion step, x_n is the input image mixed with Gaussian noise at diffusion step n, and y^{bg}, y^{ar}, and y^{ext} are background, autoregressive, and external control, respectively. Our training objective is to model the probability

$$p_{\theta_k}(\boldsymbol{X}^k_{t:t+b_k} | \boldsymbol{X}^{<k}_{t:t+b_k}, \boldsymbol{X}^{\leq k}_{<t}, \boldsymbol{Z}^k_{t:t+b_k}) \tag{6.3}$$

under the loss function

$$\mathcal{L}(\theta_k) = \mathop{\mathbb{E}}_{\boldsymbol{X},t} \ell_{\theta_k}(\boldsymbol{X}^k_{t:t+b_k}, \boldsymbol{X}^{<k}_{t:t+b_k}, \boldsymbol{X}^{\leq k}_{<t}, \boldsymbol{Z}^k_{t:t+b_k}), \tag{6.4}$$

where

$$\ell_{\theta_k}(x, y^{\text{bg}}, y^{\text{ar}}, y^{\text{ext}}) = \mathbb{E}_{\epsilon,n}||\epsilon - \epsilon_{\theta_k}(x_n, n, y^{\text{bg}}, y^{\text{ar}}, y^{\text{ext}})||_2^2. \tag{6.5}$$

Algorithm 1 Whole-song generation algorithm.

Constants: Resolution factor for each level $r_1 = 1, r_2 = \gamma, r_3 = r_4 = \delta\gamma$
Input: External control $\boldsymbol{Z}^k (2 \leq k \leq 4)$ (optional)

```
1: $\boldsymbol{X}^1 \sim p_{\theta_1}(\cdot|\emptyset, \emptyset, \emptyset)$
2: $M \leftarrow \textsc{InferSongLength}(\boldsymbol{X}^1)$
3: for $k = 2, \ldots, 4$ do
4:     $\boldsymbol{X}^k_{0:h_k} \sim p_{\theta_k}(\cdot|\boldsymbol{X}^{<k}_{0:b_k}, \emptyset, \boldsymbol{Z}^k_{0:b_k})$
5:     for $t = 0, h_k, 2h_k, \ldots, r_k M - b_k$ do
6:         $\boldsymbol{X}^k_{t+h_k:t+b_k} \sim p_{\theta_k}(\cdot|\boldsymbol{X}^k_{t:t+h_k}; \boldsymbol{X}^{<k}_{t:t+b_k}, \boldsymbol{X}^{\leq k}_{<t}, \boldsymbol{Z}^k_{t:t+b_k})$
7:     end for
8: end for
9: return $\{\boldsymbol{X}^k | 1 \leq k \leq 4\}$
```

6.2.4 Whole-Song Generation Algorithm

At the inference stage, we leverage the conditional probability (see Equation 6.3) to achieve whole-song generation by autoregressively inpainting the generated segments. Inpainting is a commonly-used method in diffusion models for image editing, and is developed as a quasi-autoregressive method for sequential generation [174]. In our algorithm, we use a hop length of $h_k := b_k \mathbin{/\!/} 2$ for inpainting, and the algorithm is shown in Algorithm 1. Note that in training, we zero-pad $\boldsymbol{X}^1$ to 256 measures, so during inference, we derive the actual song length by finding the first all-zero entries of the generated $\boldsymbol{X}^1$. This process is denoted by InferSongLength($\cdot$). Here, we use

$$\boldsymbol{X}^k_{t+h_k:t+b_k} \sim p_{\theta_0}(\cdot|\boldsymbol{X}^k_{t:t+h_k}; \boldsymbol{X}^{<k}_{t:t+b_k}, \boldsymbol{X}^{\leq k}_{<t}, \boldsymbol{Z}^k_{t:t+b_k}) \tag{6.6}$$

to indicate the distribution of the second half of the sequence conditioned on the first half via inpainting, together with the background, autoregressive, and external conditions.

6.3 Analysis of Structural Music Generation

In this section, we show an example of whole-song music generation of **40 measures** in Figure 6.2. The given Form of the piece has a simple verse-chorus structure with 4-measure verse and 8-measure chorus phrases appearing multiple times.

The generated music shows a clear music structure. The melodies of three verses all consist of syncopated rhythm in a narrow pitch range, while the melodies of two choruses are both relatively lyrical with a broader pitch range (indicated by shaded rectangles). The accompaniment pattern predominantly features eighth notes in verses and sixteenth notes in choruses. Moreover, the cadences at phrase boundaries are clearly indicated by the tonic or dominant chords and the "fill" in the accompaniment (indicated by dotted red rectangles). Furthermore, we notice the music intensity in the second half is stronger than in the first half, realized by more active pitch and rhythm movements and higher pitches (indicated by shaded rectangles with dotted borders). Such intensity changes make the composition go to a climax point before ending, showing a well-formed chronological structure.

In Appendix C.3, we break down the hierarchical generation process and show examples of structural controllability of each level. More generation results are available at the demo page.

6.4 Experiments

We focus our experiments on the generation of Lead Sheet and Accompaniment, the two lower levels of languages. The rationale is that the information of higher-level languages is difficult to evaluate directly, and they are implied at the lower levels. In this section, we first evaluate the

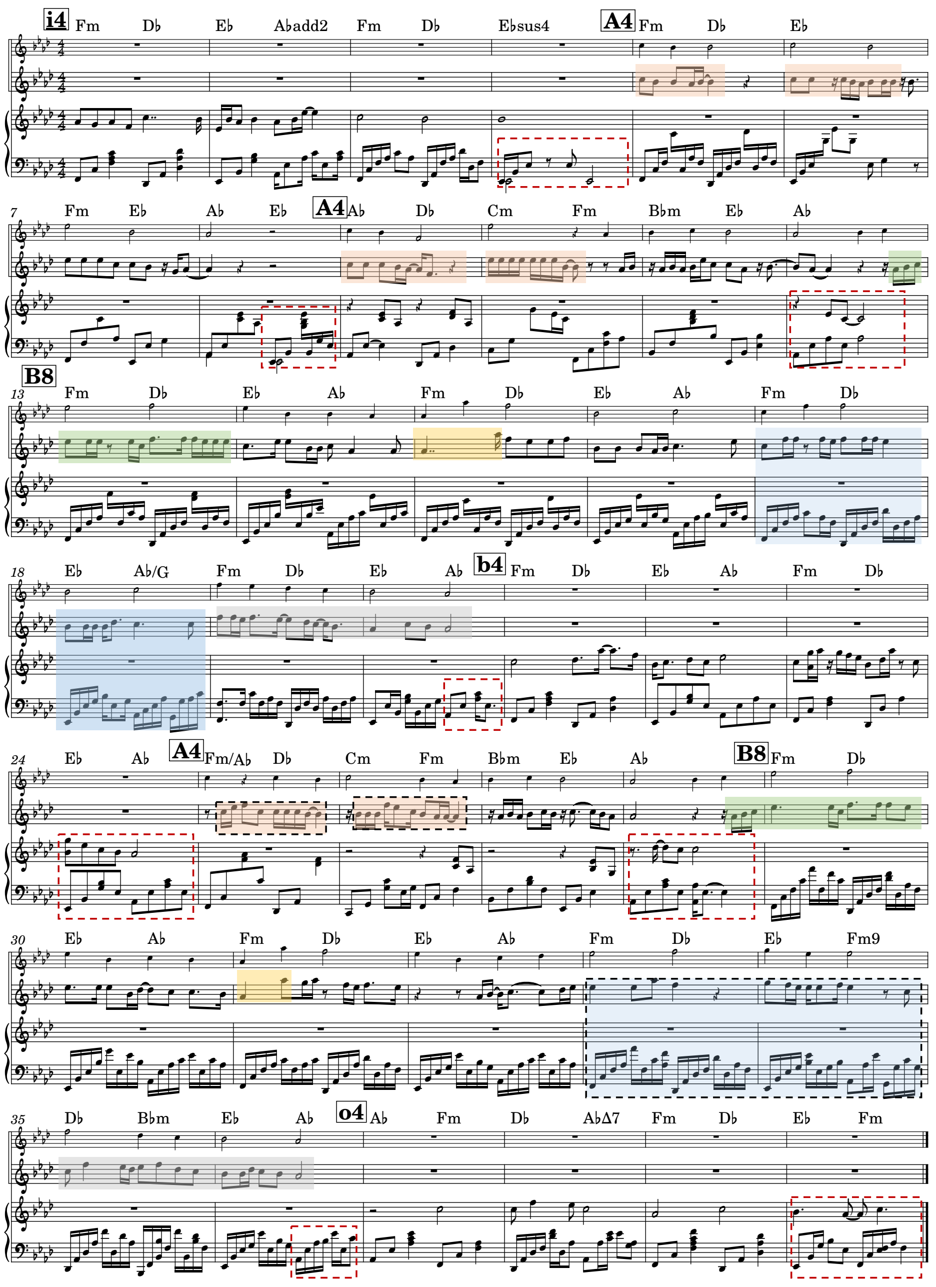

i4 Fm D♭ E♭ A♭add2 Fm D♭ E♭sus4 A4 Fm D♭ E♭
7 Fm E♭ A♭ E♭ A4 A♭ D♭ Cm Fm B♭m E♭ A♭
B8 13 Fm D♭ E♭ A♭ Fm D♭ E♭ A♭ Fm D♭
18 E♭ A♭/G Fm D♭ E♭ A♭ b4 Fm D♭ E♭ A♭ Fm D♭
24 E♭ A♭ A4 Fm/A♭ D♭ Cm Fm B♭m E♭ A♭ B8 Fm D♭
30 E♭ A♭ Fm D♭ E♭ A♭ Fm D♭ E♭ Fm9
35 D♭ B♭m E♭ A♭ o4 A♭ Fm D♭ A♭Δ7 Fm D♭ E♭ Fm

Figure 6.2: An example of whole-song generation of 40 measures under a given Form (A♭ major key and "i4A4A4B8b4A4B8o4" phrases). The three staves (from top to bottom) show the generated Reduced Lead Sheet, Lead Sheet, and Accompaniment. Here, rectangles with colored background are used to indicate the appearance of the same motifs in verse and chorus sections. Dashed border rectangles with colored background indicate a variation of motifs. We use red dotted rectangles to show where the generated score shows a strong implication of phrase boundary or cadence. The generated chord progressions in Reduced Lead Sheet and Lead Sheet are identical, shown by the chord symbols.

structure of full pieces based on a proposed objective metric, and then subjectively evaluate both structure and quality on music segments (8-measure) and whole-song samples (32 measures). For more evaluations, we refer the readers to Appendix C.

6.4.1 Dataset

We use the POP909 dataset to train our model [250]. POP909 is a pop song dataset of 909 MIDI pieces containing lead melodies, secondary melodies, piano accompaniment tracks, key signatures, and chord annotations. We pad each song to 256 measures to train Stage 1, and segment each song into corresponding time scopes (128 beats for Stage 2 and 128 steps for Stage 3 and 4) with a hop size of one measure. 90% of the songs are used for training and the rest 10% are used for testing. Training samples are transposed to all 12 keys. In Appendix C.1, we introduce more details on data processing.

6.4.2 Baseline Settings

We construct two baseline models for whole-song Lead Sheet and Accompaniment generation tasks. The two models are modified from two state-of-the-art phrase-level generation models: a diffusion-based one and a transformer-based one.

Diffusion-based (*Polyff.+ph.l.*). We add phrase label control to Polyffusion [174] as the external condition, and use the iterative inpainting technique to generate full pieces. We train two separate diffusion models for Lead Sheet and Accompaniment generation on POP909, both adopting the same data representations as the proposed method. This also serves as an ablation study on the effectiveness of our cascaded model design.

Transformer-based (*TFxl(REMI)+ph.l.*). [180] enables phrase label control on the REMI representation [108] with Transformer-XL as the model backbone. Similarly, we train two versions for Lead Sheet and Accompaniment generation on POP909.

6.4.3 Evaluation

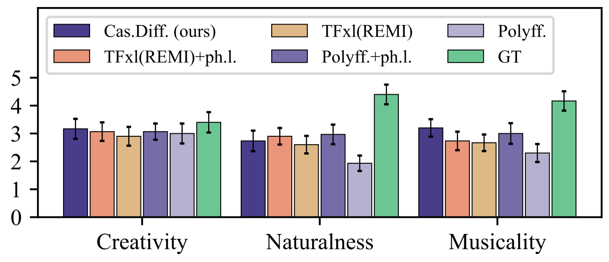
Objective evaluation. For whole-song well-structuredness, we design the *Inter-Phrase Latent Similarity* (ILS) metric to measure music structure based on content similarity. The metric encourages similarity between music with same phrase labels (e.g., two verses), and distinctiveness between music with different phrase labels (e.g., verse and chorus). We leverage pre-trained disentangled VAEs that encode music notes into latent representations and compare cosine similarities in the latent space. Given a similarity matrix showing pairwise similarity of 2-measure segments within a song, ILS is defined as the ratio between the average similarity between phrases of the same type and the average similarity between all phrases. Therefore, higher values indicate better structure.

	Lead Melody		**Chord**	**Accompaniment**
	ILS$^{\mathrm{p}}$ ↑	ILS$^{\mathrm{r}}$ ↑	ILS$^{\mathrm{chd}}$ ↑	ILS$^{\mathrm{txt}}$ ↑
Ground Truth	2.28 ± 0.14	2.30 ± 0.13	1.42 ± 0.07	1.68 ± 0.09
Cas.Diff. (ours)	$\mathbf{2.05} \pm 0.14$	1.49 ± 0.07	$\mathbf{1.32} \pm 0.05$	$\mathbf{1.19} \pm 0.06$
Polyff. + ph.l.	0.60 ± 0.12	0.76 ± 0.05	0.52 ± 0.06	0.61 ± 0.04
TFxl(REMI) + ph.l.	1.89 ± 0.15	$\mathbf{1.71} \pm 0.13$	0.68 ± 0.06	0.74 ± 0.04

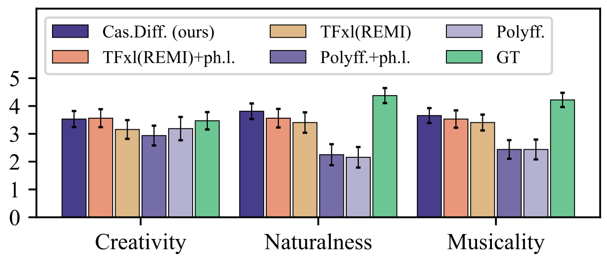
Table 6.2: Objective evaluation of music structure via the proposed inter-phrase latent similarity.

We compute ILS on lead melody, chord, and accompaniment. Using pre-trained VAEs from [271] and [247], we compute the latent representations of pitch contour and rhythm (i.e., $z_{\mathrm{p}}, z_{\mathrm{r}}$) for lead melody, latent z_{chd} for chord, and latent texture z_{txt} for accompaniment. We predefine four types of common phrases and let models generate 32 samples for each phrase type, resulting in 128 full songs in total. ILS$^{\theta}, \theta \in \{\mathrm{p}, \mathrm{r}, \mathrm{chd}, \mathrm{txt}\}$ are calculated for each song, and we show their mean and standard deviation in Table 6.2. The results show our model significantly outperforms baselines on the phrase content similarity of chord and accompaniment, indicating its effectiveness in preserving long-term structure.

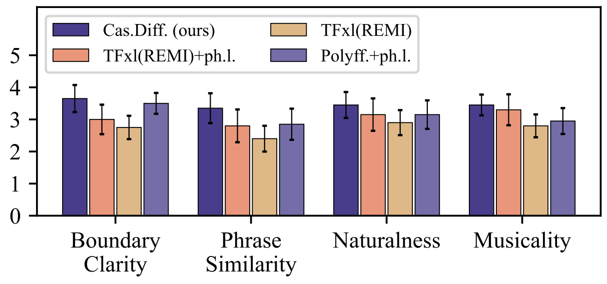
Subjective evaluation. We design a double-blind online survey that consists of two parts: a short-term (8 measures) evaluation of music quality, and a whole-song (32 measures) evalua-



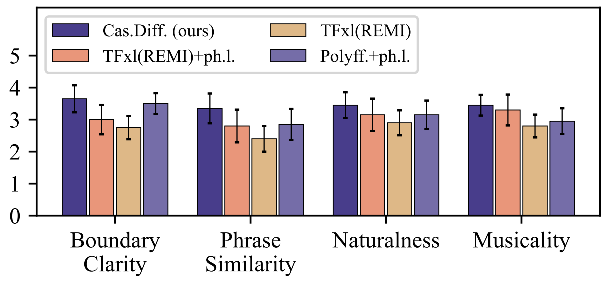
(a) Short-term lead sheet generation.



(b) Short-term accompaniment generation.



(c) Whole-song lead sheet generation.



(d) Whole-song accompaniment generation.

Figure 6.3: Subjective evaluation results on music quality and well-structuredness. GT indicates ground truth samples composed by humans.

tion of both music quality and well-structuredness. Participants rate *Creativity*, *Naturalness*, and *Musicality* for short-term music segments. For whole-song evaluation, we drop *Creativity* but introduce two more criteria: *Boundary Clarity* and *Phrase Similarity* to focus on the structure of the generation. All metrics are rated based on a 5-point scale. We constrain whole-song length to 32 measures so that the participants can better memorize the samples and the survey can have a reasonable duration. These generated pieces still preserve a condensed pop-song structure by specifying Form to contain intro, outro, and repetitive verses or choruses.

Additionally, for short-term evaluation, we use two more reference models: *Polyff.* model and *TFxl(REMI)* model, two baseline models without phrase label conditions. This is to investigate whether the introduction of phrase control causes degradation in music quality. For each model, we select three samples for both short-term and whole-song evaluation as well as both lead sheet and accompaniment generation, resulting in $3 \times 2 \times 2 = 12$ groups of samples. Each group of samples shares the same prompt (2 measures for 8-measure samples, and 4 measures for 32-

measure samples) and phrase labels (for whole-song evaluation). In the survey, both the group order and the sample order are randomized.

A total of 57 people participated in our survey, and the evaluation result is shown in Figure 6.3. The bar height shows the mean rating, and the error bar shows the 95% confidence interval (computed by within-subject ANOVA). We show that our model significantly outperforms baselines in the structural metrics of whole-song generation, especially in accompaniment generation. Our model consistently outperforms baselines in terms of music quality in both short-term and whole-song generation, proving that the introduction of compositional hierarchy does not hinder the generation quality.

6.5 Related Work

In this section, we first review music structure in musicology in Section 6.5.1, followed by music structure modeling in deep music generation approaches in Section 6.5.2. Finally, in Section 6.5.3 we review the state-of-the-art deep generative methods relevant to the problem of whole-song generation.

6.5.1 Music Structure Modeling

Traditional music theory focuses on the analysis of music structure in terms of counterpoint [49], harmony [211], forms [127], etc. In the early 20th century, a more comprehensive theory, *Schenkerian analysis* [209], emerged with a focus on the *generative* procedure of music. The theory introduces a compositional hierarchy of music, aiming to show how a piece is composed from its *background*, the normal form of music, to its *middle ground*, where music form and rough music development are realized, and finally to the *foreground*, the actual composition.

Nowadays, compositional hierarchy is still prevalent in modern musicology. Notable developments include [219], a general compositional hierarchy for pop music, and *Generative Theory* *of Tonal Music* (GTTM) [141], a theory focusing on the definition and analysis of formal musical syntax. From a computational perspective, these studies provide more formal music features and computer-friendly generative processes [87, 88]. The focus of this chapter is to further leverage the compositional hierarchy of music to develop a fully computable language and to model it with deep neural networks.

6.5.2 Structured Deep Music Generation

Recent advances in deep generative models have greatly improved music generation quality, primarily by more effective modeling of the local musical structure in two ways: implicit and explicit. Implicit approaches, exemplified by models such as Music Transformer [106], Muse-BERT [244], and Jukebox [59], learn structures by predicting and filling musical events, often revealing context dependencies via attention weights. Explicit approaches leverage domain knowledge to define music features or extract interpretable music representations, allowing the learning of structures like measure-level pitch contour and accompaniment [57, 247, 271, 287]. This study aims to combine both explicit and implicit approaches and further model phrase and whole-song structures. The explicit modeling lies in our definition of a computational hierarchical music language, and the implicit modeling of the structure lies in the cascaded diffusion models.

6.5.3 Diffusion and Cascaded Modeling for Music Generation

Diffusion models, after their success in image and audio domains, have very recently been applied to music generation [145, 174, 175]. Besides high sample quality, diffusion models naturally lead to coherent local structures with the innate inpainting method [161], i.e., by generating music segments conditioned on surrounding contexts. As for long-term structures, we recently saw the design of cascaded diffusion modeling in Moûsai [210], which generates high-fidelity audios using multi-scale sampling.

In this study, our focus is on symbolic music, and we adopt the idea of multi-scale generation with cascaded models. Additionally, we integrate the cascaded process with the proposed hierarchical music language so that each layer of the diffusion model focuses on a certain interpretable aspect of music composition. In particular, all levels of music languages are defined as image-like representations. Inspired by sketch- and stroke-based image synthesis [40], we model hierarchical music generation by regarding high-level and low-level music languages as the background and foreground "strokes", respectively.

6.6 Summary

In summary, we contribute the first hierarchical whole-song deep generative algorithm for symbolic music. The current study focuses on the pop music genre, and experimental results demonstrate that our model consistently generates more structured, natural, and musical outputs compared to baseline methods, both at the whole-song and the phrase scales. Additionally, our model offers extensibility, allowing flexible external controls via pre-trained music embeddings. Our approach relies on two key components: a hierarchical music language that balances human interpretability with computational tractability, and a cascaded diffusion architecture that effectively captures the hierarchical structure of entire compositions through both top-down and context-dependent mechanisms. It demonstrates that a strong structural inductive bias can lead to more effective and efficient learning for deep music generative models, and such methodology is potentially useful for other domains as well. We see this study shedding light on future directions, such as extending our hierarchical language and generation approach to multi-track symbolic music and audio-based music generation.

7 | Concept Hierarchy

This chapter continues to explore the other challenge of concept organization: the *concept hierarchy*. While compositional hierarchy has been commonly explored in existing works [59, 137, 203], concept hierarchy has received far less attention. It involves generating abstract ideas, transforming them, and organizing them meaningfully across musical time and space to form a complete composition. This chapter investigates the problem in the context of full-song electronic music generation, where musical ideas are often associated with audio or MIDI samples from a library, making them more tractable to manipulate and transform into a complete composition.

This chapter is based on the published work *TOMI: Transforming and Organizing Music Ideas for Multi-Track Compositions with Full-Song Structure* [94], conducted in collaboration with Qi He and Gus Xia.[1]

7.1 The TOMI Framework

Automatic music generation has advanced from producing short clips to composing entire pieces, yet long-term structure remains a major challenge. Unlike short-term generation, which focuses on capturing local patterns [3, 52, 71, 83, 109, 174], long-term generation requires handling structure across multiple levels, from sectional repetition and cadence to the overall theme and whole narrative flow. The most common approach is to scale up models and data [59, 276],

[1]The code is available at `https://github.com/heqi201255/TOMI`, and the demo page can be accessed via `https://tomi-2025.github.io/`.

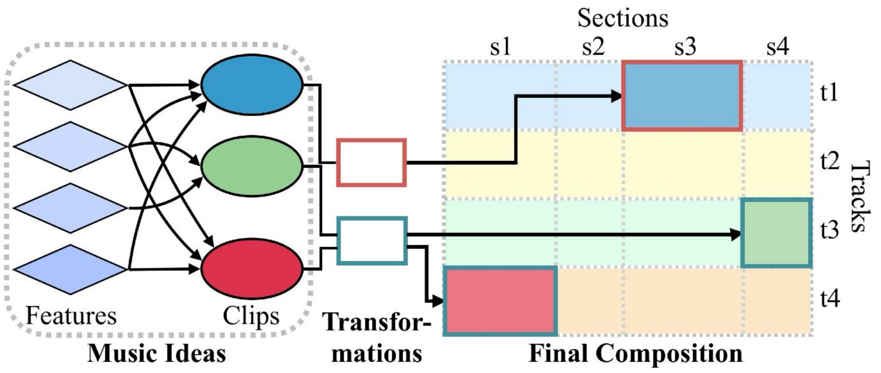


Figure 7.1: Concept hierarchy in TOMI: music ideas developed from features to clips are transformed and integrated into the composition, organized by sections and tracks.

yet even with vast training corpora, achieving truly structured compositions remains difficult. An alternative is to model the *temporal hierarchy* of music, where it evolves at multiple time scales, with different model components capturing context dependencies at each level [57, 59, 203, 243].

But have we fully captured the hierarchical nature of music? While temporal hierarchy is essential, so is *concept hierarchy*, which often manifests in the development of motifs and music materials. As noted in [54], most pop songs are composed with a sparse and small set of core ideas, which are then evolved and repeated in an organized way. We illustrate this process in Figure 7.1, where music ideas, initially appearing as abstract features, are *concretized* as music clips, then *transformed*, and finally *organized* into a full composition. We refer to this concept hierarchy in music as TOMI (**T**ransforming and **O**rganizing **M**usic **I**deas). TOMI shares a spirit with David Cope's recombinant music composition in EMI [51], an idea that deep learning has yet to truly embrace.

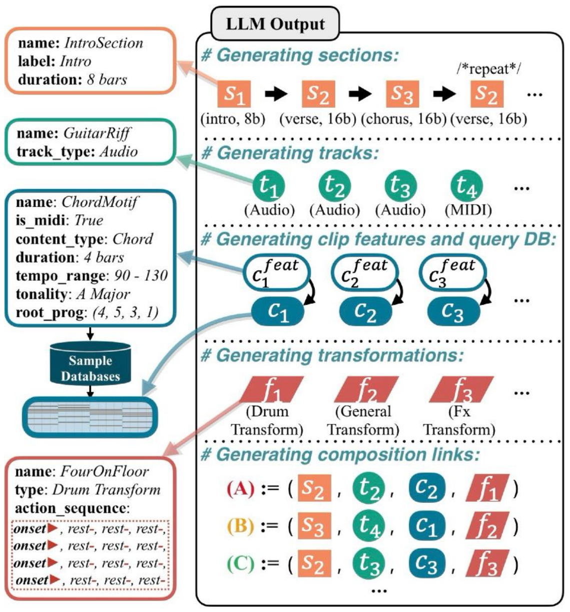
In this chapter, we propose a TOMI-based music generation system for full-song and multi-track electronic music composition, inspired by the prevalent use of MIDI and audio sample packs among electronic music producers. The system is built around four key elements: *clips*, *transformations*, *sections*, and *tracks*. Sections and tracks are first created to serve as the canvas for composition, similar to a digital audio workstation (DAW) interface. The system then selects music clips from a library of audio and MIDI samples. For each selected clip, a transformation function is defined and applied, then the transformed clip is placed in its designated section and track. On the backend, a structured data representation parameterizes clips, transformations, sections, and tracks, and links them dynamically to construct a full composition. We leverage a pre-trained text-based large language model (LLM) to operate on this data structure, using in-context learning (ICL) to fill in the parameters and create dynamic links.

In sum, the contributions of this chapter are as follows:

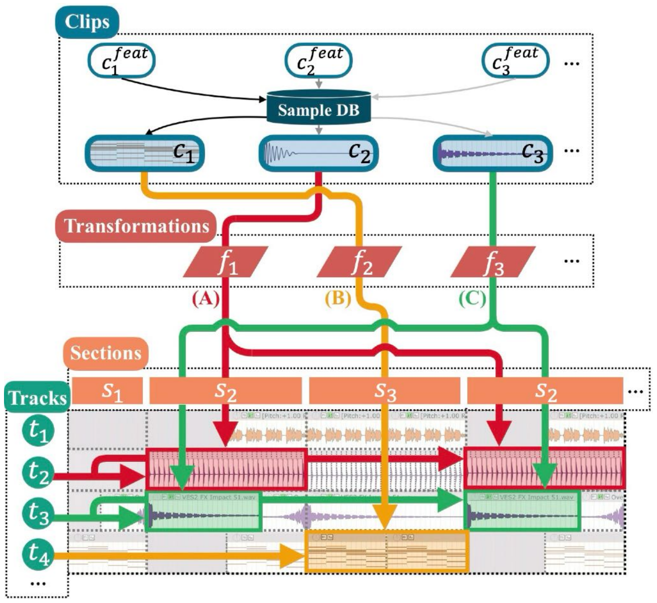
1. **We introduce TOMI to model music concept hierarchy** and develop a deep learning-based system for structured electronic music generation. The proposed data structure integrates symbolic and audio representations and can be manipulated by text-based LLMs via ICL.
2. **We apply our system to generate high-quality electronic music with full-song structure**. Objective and subjective evaluations show that songs generated by our model have clearer phrase boundaries, better phrase development, and higher music quality than the baselines.
3. **We integrate TOMI with the REAPER digital audio workstation**, providing seamless connection with professional music software interface and enabling human-AI co-creation with high-resolution audio rendering.

7.2 Methodology

In this section, we discuss TOMI in multi-track electronic music generation with full-song structure. The implementation consists of two main components: (1) a graph data structure named *composition link* that connects raw music ideas with the composition (Section 7.2.1), and (2) in-context learning to compose music by following this data structure (Section 7.2.2). We also demonstrate the integration with the REAPER DAW (Section 7.2.3).


LLM Output
# Generating sections:
/*repeat*/
(intro, 8b) (verse, 16b) (chorus, 16b) (verse, 16b)
# Generating tracks:
(Audio) (Audio) (Audio) (MIDI)
# Generating clip features and query DB:
# Generating transformations:
(Drum Transform) (General Transform) (Fx Transform)
# Generating composition links:
(A) := (s2, t2, c2, f1)
(B) := (s3, t4, c1, f2)
(C) := (s2, t3, c3, f3)
name: IntroSection
label: Intro
duration: 8 bars
name: GuitarRiff
track_type: Audio
name: ChordMotif
is_midi: True
content_type: Chord
duration: 4 bars
tempo_range: 90 - 130
tonality: A Major
root_prog: (4, 5, 3, 1)
Sample Databases
name: FourOnFloor
type: Drum Transform
action_sequence:
onset, rest-, rest-, rest-,
onset, rest-, rest-, rest-,
onset, rest-, rest-, rest-,
onset, rest-, rest-, rest-

(a) Example of an LLM output following our ICL prompt structure. In clips, the LLM only generates features for clips, which are then used to query the sample databases for actual MIDI or audio clips. We also show three composition links, each as a tuple of (section, track, clip, and transformation).



(b) Music generation process with composition links, illustrating how clips (top) are transformed and then organized into the arrangement (bottom). Links (A) and (C) each have two branches because section s_2 appears twice in the section sequence, which means they are identical and share the same composition links.

Figure 7.2: We show the structured LLM output in (a) with the corresponding music generation process in (b). We distinguish node types by color and shape, with a detailed attribute example for each type shown on the left of (a). We depict three composition link examples as colored arrows in (b), highlighting their results with rectangles in corresponding colors.

7.2.1 TOMI Data Structure

The proposed data structure consists of four node types: *clips*, *sections*, *tracks*, and *transformations*. A *composition link* is a quadruple of these nodes, specifying a music clip (*what*) to be placed in a particular section (*when*) and on a specific track (*where*), undergoing certain transformations (*how*). Nodes are reusable across links, forming a structured representation of the full composition.

Clip Node

Clips are audio or MIDI samples sourced from databases. Each clip is a short music segment, such as a chord progression or a drum loop, described by a set of features. A MIDI clip can represent elements like chords, basslines, melodies, or arpeggios, with attributes such as tonality, duration, and root progression. An audio clip can be either a sample loop or a one-shot sound, with keywords describing its content. The blue-edged box in Figure 7.2(a) shows an example feature set for a chord clip. Once the features are specified by the LLM (discussed in Section 7.2.2), we can query the databases for the corresponding music material.

Section Node

The temporal divisions of a music composition are modeled by section nodes. A section node involves a duration and a phrase label, such as verse and chorus. A section node can appear multiple times within a composition, meaning its music content remains identical across instances. For example, as shown in Figure 7.2, the 16-bar verse section s_2 is reused after the 16-bar chorus section s_3, resulting in two identical verses with the same content.

Track Node

Track nodes represent the vertical layering of a composition and organize clips into MIDI or audio tracks (see the track axis in Figure 7.2(b)). MIDI tracks accept only MIDI clips and require instruments to generate sound, while audio tracks accept only audio clips and play them directly.

Transformation Node

A transformation node transforms clips before placing them on specific tracks and sections. Unlike pitch transposition and tempo adjustment, which are handled in the final stage (discussed in Section 7.2.3), transformation nodes perform more semantically meaningful manipulations and mainly serve three roles. First (and most importantly), it can control rhythmic patterns of the audio or MIDI clips with an *action sequence* of onsets, sustains, and rests (see f_1 in Figure 7.2(a)). For example, it can convert MIDI chord clips with whole notes into rich rhythmic patterns or place a one-shot drum on each beat to create a four-on-the-floor rhythm. Second, it can handle the riser or faller sound effects commonly used in electronic music composition, which are placed either left- or right-aligned within a section, creating smooth transitions (see c_3 in Figure 7.2(b)). Lastly, it dynamically handles the looping of a clip, determining whether it plays once at a specific time, loops throughout a section, or is trimmed to fit a shorter section. Our implementation defines three subclasses of transformations: (1) Drum transform (for one-shot drums), (2) Fx transform (for risers and fallers), and (3) General transform (for others), each following different transformation rules. We refer the reader to our demo page for details.

Composition Link

A composition link comprises one node each of section, clip, transformation, and track, showing the entire process of transforming and organizing a music idea in the composition. Since nodes are independent of the composition link, they can be reused across multiple links. Figure 7.2(b) shows the process of three composition links. Note that links (A) and (C) branches in the final composition because section s_2 is used twice. This is an efficient way to represent complex arrangements, as a single clip can be referenced by multiple links, spanning different sections and tracks while adapting to various transformations.

7.2.2 Music Generation with In-Context Learning

The TOMI data structure, as defined above, can be fully represented in text format. A complete composition can be decomposed into a set of composition links, each represented as a quadruple of nodes, where each node corresponds to a set of textual attributes. By leveraging a text-based LLM with in-context learning, we can generate compositions directly as TOMI instances. Our

ICL prompt systematically breaks down the data structure in steps, following the order: *sections*, *tracks*, *clips*, *transformations*, and *composition links*. It guides the LLM from planning the overall song structure and music ideas to organizing them into a full song. The LLM output follows these steps accordingly, as shown in Figure 7.2(a).

In our implementation, we elaborate our data structure with detailed examples and define a structured response schema in the prompt. We require the LLM to generate a unique *name* attribute for each node to reference in composition links. Moreover, we can specify additional contexts, such as tempo, mood, and custom song structures. To get robust results, we implement a rule-based validation to check the LLM output for syntax errors and invalid values. If issues are detected, an error report is generated, prompting the LLM to iteratively refine its output until no errors remain. At this point, we obtain an abstract representation of the composition in TOMI structure. To realize this, we initiate the sample retrieval process to get the actual clip materials. Then, we set a global *tempo* and *key* to unify the keys and tempos of clips within the DAW.

7.2.3 Digital Audio Workstation Integration

To support audio rendering and interactivity, we integrate the TOMI framework with the REAPER DAW. This allows the generated composition to be visualized in a professional DAW while benefiting users from REAPER's editing and rendering capabilities. The composition links and nodes of a generated piece are converted to REAPER elements, including tracks, section markers, and clips with applied transformations. We use REAPER to automatically time-stretch loopable clips to fit the *tempo* setting and transpose melodic clips to align with the *key* setting.

7.3 Experiment

To implement the generation system, we prepare a MIDI database and an audio database for clip sample retrieval and use GPT-4o [185] to generate compositions in the TOMI schema. We evaluate our approach with baseline methods and use both objective and subjective measurements to compare the music quality and structural consistency.

7.3.1 System Preparation

We collect multiple licensed MIDI and audio sample packs in the electronic music genre through online purchases.[2] We process the raw datasets into separate MIDI and audio databases using different feature extraction methods. For MIDI clips, we developed a script to analyze and extract musical features from them. Moreover, it can also extract music stems, such as bass, chord, and melody, from the source MIDI to augment the data. Then, we store the labeled data in a SQLite3 [99] database as our MIDI database. For audio samples, we use ADSR Sample Manager [2] to analyze and generate labels for them. The results are also exported as a SQLite3 database. The statistics of our MIDI database and audio database are shown in Table 7.1. Sample retrieval involves constructing a search query based on the clip's attributes to fetch matching samples from the database. The clip node randomly selects one sample from the results. If no matches are found, the clip and its associated composition links are discarded.

We limit all generated sections to the 4/4 time signature to simplify implementation. When exporting audio via REAPER, we randomly assign each MIDI track to one of eight virtual instrument presets (5 for chords, 2 for melody, and 1 for bass). We keep all REAPER settings at their defaults and apply no mixing plug-ins except for a limiter on the master track to prevent audio clipping.

7.3.2 Baseline Method and Ablations

We compare our approach with MusicGen [52] and two ablations in electronic music generation. Since our system integrates both audio and MIDI data, there is no directly comparable baseline. However, as our final output is rendered as audio, MusicGen is a suitable comparison,

[2]We obtain sample packs from two music asset platforms: https://splice.com/ and https://www.loopmasters.com/.

Content Type	Count
Chord	2604
Bass	209
Melody	1392
Arpeggio	227
Total	**4432**

(a) MIDI content types.

Duration	Count
4-bar	2947
8-bar	1417
16-bar	68
Total	**4432**

(b) MIDI durations.

Sample Type	Count
Loop	104493
One-Shot	170187
Total	**274680**

(c) Audio sample types.

Loop Duration	Count
2-bar	24922
4-bar	27214
8-bar	24638
16-bar	27719
Total	**104493**

(d) Audio loop durations.

Table 7.1: Statistics of sample databases.

which is capable of generating long-term and high-quality electronic music. This allows us to evaluate our method against state-of-the-art music generation systems. The ablations help us assess the individual contributions of the composition link representation and the LLM integration.

MusicGen We use the MusicGen-Large-3.3B model as the baseline with prompts specifying key, tempo, and section sequence. To generate longer audio beyond its duration limit, we apply a sliding window approach, where a 30-second window slides in 10-second chunks, using the previous 20 seconds as context. We modify its inference process to include the current generation's position within the full composition and its corresponding phrase notations in the prompt at each step, guiding the model to align its output with the given structure.

Standalone LLM (TOMI w/o Composition Links) We remove the composition links representation from our system. We redesign the prompt to let the LLM generate a sequence of tracks and clip descriptions with position information (time point and track location) conditioned on a section sequence. The sample retrieval mechanism is also applied for clips.

Random (TOMI w/o LLM) We replace the LLM operations in our system with a rule-based method that uses randomized operations to generate music within the composition-links struc-

Method	$\text{FAD}^{\text{VGGish}} \downarrow$	$\text{FAD}^{\text{CLAP}} \downarrow$	$\text{ILS}^{\text{MERT}} \uparrow$	$\text{ILS}^{\text{MS}} \uparrow$	$\text{ILS}^{\text{WF}} \uparrow$
TOMI	**3.51**	**0.38**	**0.28** $\pm$ 0.12	**0.36** $\pm$ 0.33	**1.14** $\pm$ 0.73
MusicGen	5.31	0.62	0.06 $\pm$ 0.04	0.12 $\pm$ 0.07	0.28 $\pm$ 0.09
Standalone LLM	5.84	0.46	0.16 $\pm$ 0.11	0.10 $\pm$ 0.12	0.09 $\pm$ 0.16
Random	6.92	0.47	0.16 $\pm$ 0.09	0.22 $\pm$ 0.16	0.48 $\pm$ 0.28

Table 7.2: Objective evaluation results of FAD with two models and ILS with three latent representations.

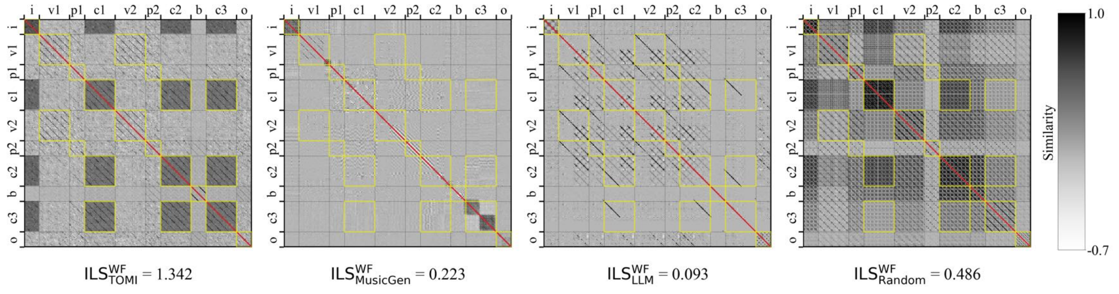


Figure 7.3: ILS similarity matrices example, where all four compositions are generated under the section sequence: *i* (intro), *v1* (verse 1), *p1* (pre-chorus 1), *c1* (chorus 1), *v2* (verse 2), *p2* (pre-chorus 2), *c2* (chorus 2), *b* (bridge), *c3* (chorus 3), and *o* (outro). Darker colors indicate higher segment similarity. The blocks marked as yellow-edge boxes are same-label similarities, with diagonal values marked as red lines being excluded, and the remaining parts are different-label similarities.

ture. The system creates 15–25 track nodes, populates sections with clips stochastically, and determines for each track whether to place, reuse, or generate a new clip. MIDI clips are assigned a random type (chord, bass, or melody) with bass derived from chords, while audio clips are selected from tonal, percussion, and sound effect feature labels. Each clip is then linked to one of four predefined transformations: general, drum, riser Fx, or faller Fx.

We define 4 keys and 4 distinct section sequences, each consisting of 8 to 10 sections. Each section has a name, a phrase label, and a duration ranging from 4 to 16 bars. Then, using each method, we generate 4 sets of electronic music pieces at 120 BPM, with each set containing 8 pieces (2 per key), all conditioned on the same section sequence. In total, we generate 32 compositions for each method.

7.3.3 Objective Evaluation

We evaluate the music quality and structural consistency of generated compositions. For music quality, we use the **Fréchet Audio Distance** metric (FAD) [121, 227] with a VGGish model [96] and a CLAP model [261] to compare human-composed electronic music and the music generated by each method. We randomly collected 329 songs as references from the Spotify Mint playlist [115], one of the most popular curated playlists dedicated to the electronic music genre. A lower FAD score indicates that the generated music is closer to human-composed music in quality.

For structural consistency, we use the **Inter-Phrase Latent Similarity** metric (ILS) refined from [243]. ILS aims to compute a self-similarity matrix of musical features and evaluate if the average similarity between segments sharing the same phrase label is higher than those with different labels. Instead of measuring the ratio of same-label to overall similarities in the original metric, we use Cohen's d [50] to compute the effect size of the difference between same-label and different-label similarities, excluding diagonal elements to avoid biases. This offers a scale-independent measure that robustly captures the separation between groups. We extract latent representations from audio while preserving temporal structure, then compute a self-similarity matrix of them using cosine similarity. We evaluate ILS with MERT embeddings [148], Mel spectrograms, and raw waveforms, denoted as ILS^{MERT}, ILS^{MS}, and ILS^{WF}, respectively. To compute the ILS score, we denote the cosine similarity between time-related latent elements i and j as S_{ij}, the phrase label of element i as l_i, and the sizes of same-label and different-label elements as N_{same} and N_{diff}, respectively, excluding diagonal elements. A higher ILS score indicates better and more effective structural consistency. The formula is defined as:

$$\text{ILS} = \frac{\bar{X}_{\text{same}} - \bar{X}_{\text{diff}}}{s}, \tag{7.1}$$

where $\bar{X}_{\text{same}}$ and $\bar{X}_{\text{diff}}$ are the mean similarities of same-label and different-label groups, respectively, defined as:

$$\bar{X}_{\text{same}} = \frac{\sum_{i \neq j} S_{ij} \cdot \delta(l_i = l_j)}{N_{\text{same}}}, \tag{7.2}$$

$$\bar{X}_{\text{diff}} = \frac{\sum_{i,j} S_{ij} \cdot \delta(l_i \neq l_j)}{N_{\text{diff}}}. \tag{7.3}$$

and s is the pooled standard deviation derived from the variances s^2_{same} and s^2_{diff} of the two groups, defined as:

$$s = \sqrt{\frac{(N_{\text{same}} - 1)s^2_{\text{same}} + (N_{\text{diff}} - 1)s^2_{\text{diff}}}{N_{\text{same}} + N_{\text{diff}} - 2}}. \tag{7.4}$$

We show the FAD scores and ILS with means and standard deviations in Table 7.2. The results show that our method achieves the lowest FAD scores and the highest ILS scores across all measurements, outperforming the baseline methods in both music quality and structural consistency. Notably, the ablation results show that even with high-quality music samples, lacking sufficient arrangement logic can still lead to poor FAD scores. This proves the ability of our approach to generate music with improved flow naturalness and arrangement coherence while preserving the expected long-term structures. Figure 7.3 provides a comparative visualization of ILS matrices for four compositions, with phrase labels on both axes, yellow-edged boxes highlighting regions of same-label similarities, the red line marking the main diagonal, and the remaining areas representing different-label similarities.

7.3.4 Subjective Evaluation

We conduct a double-blind online survey to compare the music quality and structural consistency of compositions generated by the four methods. Since each composition is a full 3- to 4-minute song, which is too long to be evaluated in a single question, we assess four compositions (one per method) throughout the survey. To ensure a comprehensive evaluation, we create three

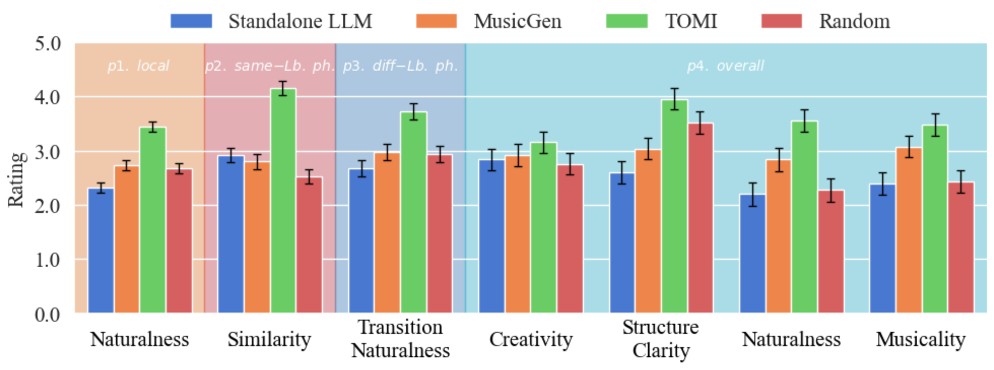


Figure 7.4: Subjective evaluation results of mean score with within-subject confidence intervals, where *p1-p4* corresponds to four parts of our survey: *p1. Local Music Quality*, *p2. Consistency Among Same-Label Phrases*, *p3. Contrast Between Different-Label Phrases*, and *p4. Overall Full-Song Evaluation*.

distinct question sets with the same survey structure but different compositions, resulting in a total of 12 compositions being evaluated. The survey is divided into four parts, each containing 4 to 12 questions, aiming to evaluate the piece from section-level music quality to overall structural alignment. All metrics are measured on a 5-point rating scale. The details are as follows:

Part 1. Local Music Quality This part consists of 3 subparts, each selecting the same section (e.g., intro) from each composition. Participants rate *Naturalness* to evaluate the similarity to human-composed music and the conformity to the typical electronic music style.

Part 2. Consistency Among Same-Label Phrases This part consists of 2 subparts, each selecting two sections with the same phrase label (e.g., verse 1 and verse 2) from each composition. Participants rate *Similarity* between the two sections.

Part 3. Contrast Between Different-Label Phrases This part consists of 2 subparts, each selecting two consecutive sections from the same position in each composition (e.g., intro and verse 1). Participants rate *Transition Naturalness* based on boundary clearness, transition smoothness, and phrase alignment.

Part 4. Overall Full-Song Evaluation This part shows the complete composition audios. Participants rate *Structure Clarity* for how well each section aligns with the given structure, *Creativity* for how creative the music is, *Naturalness* for how humanlike the music sounds, and *Musicality* for the overall music quality.

We insert an intermediate page between evaluation parts to inform participants of their progress and help reduce listening fatigue. Additionally, we show the input conditions for music generation on each question page to remind participants of the expected song structure and tonality. We distributed the survey on multiple social media platforms and received a total of 73 responses. The demographic statistics of participants are as follows:

Age ($<$18: 0%, 18-29: 69.86%, 30-44: 19.18%, 45-59: 5.48%, $\geq$60: 5.48%);

Gender (Female: 30.14%, Male: 65.75%, Non-binary: 2.74%, Prefer not to say: 1.37%);

Music background (Amateur: 34.25%, Intermediate: 41.1%, Professional: 24.66%);

Years spent on studying music (None: 26.03%, 1 year: 6.85%, 2 years: 10.96%, 3-5 years: 13.7%, 6-10 years: 16.44%, $>$10 years: 26.03%).

The subjective evaluation results are shown in Figure 7.4, where the bar height represents the mean score, and the error bar represents the confidence intervals computed by within-subject ANOVA. The results show that our method significantly outperforms the baseline in most subjective metrics, proving the effectiveness of our system in generating high-quality electronic music with solid long-term structural consistency.

7.4 Related Work

In automatic music generation, many studies focus on generating coherent music segments [3, 52, 71, 83, 109, 174], while fewer focus on modeling long-term structure under the temporal hierarchy of music. Jukebox [59] uses hierarchical VQ-VAE with time conditioning to enhance long-term coherence; Wang *et al.* [243] applies cascaded diffusion models for structured symbolic music generation. Some methods introduce structure encoding in neural networks [34, 37], employ efficient sampling techniques to generate structured variations [188], or explore multi-track joint modeling via latent diffusion [118, 119]. However, the application of TOMI concepts in music generation remains largely unexplored. Related works include a rule-based recombinant music method [51], which reorganizes existing music elements based on stylistic constraints, and MELONS [291], which uses a structure graph to enforce long-term dependencies in melody generation but offers limited flexibility in musical transformation.

In the context of co-creative music systems, existing generative models produce data in either symbolic (e.g., [83, 109, 174, 243, 291]) or waveform (e.g., [3, 52, 71]) format. Furthermore, these models lack intuitive interfaces for user co-creation and fine-grained control over musical elements, as found in modern DAWs. Composer's Assistant [168] and its successor [167] integrate with REAPER to support co-creation but focus on generating symbolic phrases rather than full pieces. Our approach generates complete compositions containing both MIDI and audio phrases while also enabling user co-creation directly within REAPER.

With recent advances in AI, text-based LLMs have emerged as a promising alternative for music creation. Previous studies like ChatMusician [275] and MuPT [200] employ ABC Notation [1] to represent music in text format and fine-tune LLaMA models [235]. While these approaches outperform generic LLMs in music tasks, their performance remains limited compared to LLMs trained exclusively on music data [106, 160, 229]. Other works have leveraged text-based LLMs for music analysis [131, 154], showing their capability to interpret music concepts through natural language. This motivates us to develop a textual hierarchical representation of music concepts to better integrate with text-based LLMs.

7.5 Summary and Future Work

We contribute TOMI, a concept hierarchy paradigm for music representation, and combine it with an ICL approach to achieve the first system for generating long-term, multi-track electronic music with both MIDI and audio clips. Experimental results show that our approach achieves high-quality generation with robust structural consistency. In addition, we integrate it with REAPER to support audio rendering and co-creation. However, harmonic coherence in our results can occasionally be disrupted by randomness and limited features during sample retrieval. Our system currently selects samples from local collections using a small feature set, which can lead to empty results or highly divergent samples. To improve this, we plan to integrate generative models for clips or use ML-based embedding models for MIDI and audio. Then, we aim to extend our model with a more sophisticated structural hierarchy to support advanced sound design and mixing. Lastly, we plan to train a TOMI-based neural network on music project files to enhance generation quality and scalability.

Part III

Concept Emergence

8 | Concept Emergence via Unsupervised Disentanglement

In Part III, we turn to the final type of concept alignment explored in this thesis: *emergence*. While Part I focused on learning implicit concepts from data, and Part II addressed how these concepts can be organized into compositional and conceptual hierarchies, the picture remains incomplete. As shown in Figure 1.1, many concepts, especially those at higher levels, are vague, unnameable, and represented as clouds. Consider, for example, the question: what governs the flow of music? This is an inherently abstract notion with no single answer. The governing concept may shift dramatically depending on the genre, context, or even personal interpretation, making it difficult to define explicitly or model directly [219].

This leads us to the notion of emergence—the ability of a model to discover and form new concepts from raw observations without relying on prior domain-specific knowledge. In this chapter, we investigate how AI systems can be equipped with inductive biases that support such emergent behavior. We begin with a basic problem: learning disentangled representations of *content* and *style* from sequences, such as distinguishing digit from handwriting style, or pitch from timbre in audio. In these settings, concepts like digits or pitches are not labeled in advance. Instead, we discover that a fundamental, domain-general inductive bias lies in a statistical asymmetry between content and style, leading to the emergence of content and style in various domains.

This chapter is based on the published work *Unsupervised Disentanglement of Content and*

Style via Variance-Invariance Constraints [262], conducted in collaboration with Yuxuan Wu, Bhiksha Raj, and Gus Xia.[1] Additional experimental details and extended discussions are provided in Appendix D.

8.1 Emergence and Content-Style Disentanglement

Learning abstract concepts is an essential part of human intelligence. Even without any label supervision, we humans can abstract rich observations with great variety into a category, and such capability generalizes across different domains and modalities. For example, we can effortlessly perceive a picture of a “cat” captured at any angle or set against any background, we can perceive the symbolic number “8” from an image irrespective of its color or writing style variations, and we can perceive an abstract pitch class “A” from an acoustic signal regardless of its timbre. These concepts form the fundamental vocabulary of our languages—be they natural, mathematical, or musical—and underpin effective and interpretable communication in everyday life.

Our goal is to emulate such abstraction capability using machine learning. We choose a *content-style representation disentanglement* approach as we believe that representation disentanglement offers a more complete picture of abstraction—concepts that matter more in communication, such as an “8” in a written phone number or a note pitch “A” in a folk song, are usually perceived as *content*, while the associated variations that often matter less in context, such as the written style of a digit or the singing style of a song, are perceived as *style*. In addition, content is usually symbolized and associated with rigid labels, as we need precise control over it during communication. E.g., to write “8” as “9” in a phone number or to sing an “A” as “B” in a performance can be a fatal error. In comparison, though style can also be described discretely, such as an “italic” writing or a “tenor” voice, a variation over it is usually much more tolerable.

[1]Demo can be found at https://v3-content-style.github.io/V3-demo/.

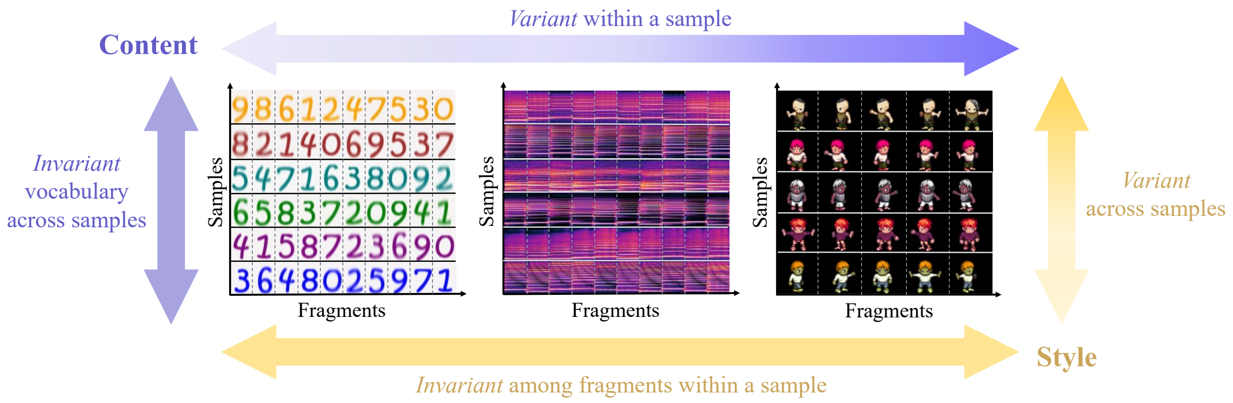

Content
Variant within a sample
Invariant vocabulary across samples
Samples
Fragments
Variant across samples
Style
Invariant among fragments within a sample

Figure 8.1: An illustration of the variance-versus-invariance constraints of content and style.

In the machine learning literature, significant progress has recently been made in content-style disentanglement for various tasks, including disentangling objects from backgrounds [101], characters from fonts [156, 265], pitch from timbre [20, 159], and phonemes from speaker identity [146, 197]. However, most existing models are either limited to specific domains [9, 147, 163] or rely heavily on domain-specific knowledge as implicit supervision. The supervision forms can be explicit content or style labels [20, 44, 120, 135, 153, 191, 192, 290], pre-trained content or style representations [197, 199], or paired data showcasing the same content rendered in different styles or vice versa [111, 206]. In addition, the disentangled representations often fell short in generalizing to new contents or styles, and they lack interpretability at a symbolic level and do not align well with human perceptions [181, 285].

To address the aforementioned challenges, achieving more generalizable and interpretable disentanglement in an unsupervised manner, we introduce V3 (***v**ariance-**v**ersus-in**v**ariance*). V3 disentangles content and style by leveraging meta-level prior knowledge about their inherent statistical differences. As shown in Figure 8.1, our design principle is based on the observation that **content and style display distinct patterns of variation**—*content undergoes frequent changes within different fragments of a sample yet maintains a consistent vocabulary across data samples, whereas style remains relatively stable within a sample but exhibits more significant variation across* *different samples.*

In this chapter, we adopt the vector-quantized autoencoder architecture and incorporate variance-versus-invariance constraints to guide the learning of latent representations that capture content-style distinctions. We demonstrate that V3 effectively generalizes across distinct areas: disentangling pitches and timbres from musical data, disentangling numbers and ink colors from images of digits, and disentangling character actions and appearances from game video clips. Experimental results show that our approach achieves more robust content-style disentanglement than unsupervised baselines, and outperforms even supervised methods in out-of-distribution (OOD) generalization and few-shot learning for discriminative tasks. Lastly, symbolic-level interpretability emerges with a near one-to-one alignment between the vector-quantized codebook and human knowledge, an outcome not yet seen in previous studies. In summary, our contributions are as follows:

1. **Unsupervised content-style disentanglement:** We introduce V3, an unsupervised method leveraging meta-level inductive bias to disentangle content and style representations, without requiring paired data, content or style labels, or domain-specific assumptions.

2. **Out-of-distribution generalization:** As a result of successful content-style disentanglement, V3 shows better out-of-distribution generalization capabilities compared to supervised methods in few-shot settings, that is, recognizing content when presented with only a few examples of unseen styles.

3. **Emergence of interpretable symbols:** Given the availability of semantic segmentations, V3 can foster the development of interpretable content symbols that closely align with human knowledge.

8.2 Methodology

Considering a dataset consisting of N data samples, where each sample contains L fragments, we aim to learn each fragment's *content* and *style* representation with the inductive bias illustrated in Figure 8.1. Intuitively, the fragments within each data sample have a relatively frequently changing content and a relatively stable style. For different data samples, the style exhibits significant variations and their content more or less keeps a consistent vocabulary. In the following, we first introduce the autoencoder architecture V3 is built upon, then the variability statistics to quantify the changing patterns of content and style, and the proposed variance-versus-invariance constraints.

8.2.1 Model Architecture

The model architecture of V3 is illustrated in Figure 8.2. Let $\boldsymbol{X} = \{\boldsymbol{x}_{ij}\}_{N\times L}$ be the dataset, where $\boldsymbol{x}_{ij}$ corresponds to the j-th fragment of the i-th sample. We use an autoencoder architecture to learn the representations of $\boldsymbol{x}_{ij}$. The encoder encodes the input data $\boldsymbol{x}_{ij}$ to the latent space, which is split into to $\boldsymbol{z}^{\mathrm{c}}_{ij}$ and $\boldsymbol{z}^{\mathrm{s}}_{ij}$. We use vector quantization as the dictionary learning method for content. Every content representation $\boldsymbol{z}^{\mathrm{c}}_{ij}$ is quantized to the nearest atom in a codebook of size K as $\tilde{\boldsymbol{z}}^{\mathrm{c}}_{ij}$. The decoder integrates $\tilde{\boldsymbol{z}}^{\mathrm{c}}_{ij}$ and $\boldsymbol{z}^{\mathrm{s}}_{ij}$ and reconstructs the fragment $\hat{\boldsymbol{x}}_{ij}$. The overall loss function is the weighted sum of three terms:

$$\mathcal{L} = \mathcal{L}_{\mathrm{rec}} + \alpha\mathcal{L}_{\mathrm{vq}} + \beta\mathcal{L}_{\mathrm{V3}}. \tag{8.1}$$

Here, $\mathcal{L}_{\mathrm{rec}}$ is the reconstruction loss of $\boldsymbol{X}$ and $\mathcal{L}_{\mathrm{vq}}$ is the VQ commit loss [184]:

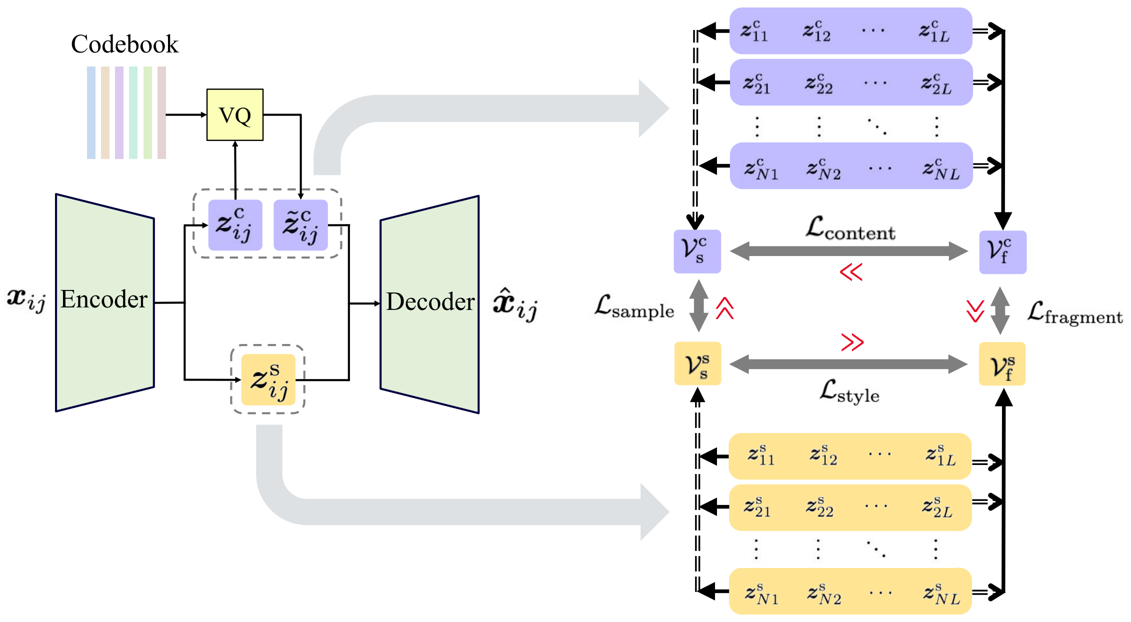

Codebook
VQ
$\boldsymbol{x}_{ij}$
Encoder
$\boldsymbol{z}^{\mathrm{c}}_{ij}$
$\tilde{\boldsymbol{z}}^{\mathrm{c}}_{ij}$
$\boldsymbol{z}^{\mathrm{s}}_{ij}$
Decoder
$\hat{\boldsymbol{x}}_{ij}$
$\mathcal{L}_{\text{content}}$
$\mathcal{L}_{\text{sample}}$
$\mathcal{L}_{\text{fragment}}$
$\mathcal{L}_{\text{style}}$
$\mathcal{V}^{\mathrm{c}}_{\mathrm{s}}$
$\mathcal{V}^{\mathrm{c}}_{\mathrm{f}}$
$\mathcal{V}^{\mathrm{s}}_{\mathrm{s}}$
$\mathcal{V}^{\mathrm{s}}_{\mathrm{f}}$

Figure 8.2: The model architecture of V3. Left: The autoencoder has two branches for content and style, respectively, where the content branch has a VQ layer at the encoder output. Right: the V3 constraints, where double-dashed arrows represent measuring the variability by $\nu_k(\cdot)$, and solid arrows represent taking the average.

$$\mathcal{L}_{\text{rec}} = \frac{1}{N \times L} \sum_{i=1}^{N} \sum_{j=1}^{L} \|\boldsymbol{x}_{ij} - \hat{\boldsymbol{x}}_{ij}\|_2, \tag{8.2}$$

$$\mathcal{L}_{\text{vq}} = \frac{1}{N \times L} \sum_{i=1}^{N} \sum_{j=1}^{L} \|\boldsymbol{z}^{\mathrm{c}}_{ij} - \mathrm{sg}(\tilde{\boldsymbol{z}}^{\mathrm{c}}_{ij})\|_2, \tag{8.3}$$

where $\mathrm{sg}(\cdot)$ is the stop gradient operation of the straight-through optimization. The final term $\mathcal{L}_{\text{V3}}$ is the proposed regularization method to ensure unsupervised content-style disentanglement, which we introduce in the rest of this section.

8.2.2 Variability Statistics

We define four statistics to measure the *degree of variability* in accordance with the four edges of Figure 8.1. These statistics are based on a backbone variability measurement $\nu_k(\cdot)$, where k

represents the dimension along which variability is computed. In this chapter, we define $\nu_k(\cdot)$ as the mean pairwise distance (MPD). Formally, for a vector $\boldsymbol{z}$ of length D,

$$\nu_{i=1}^{D}(\boldsymbol{z}_i) := \mathrm{MPD}_{i=1}^{D}(\boldsymbol{z}_i) = \frac{1}{D(D-1)} \sum_{i=1}^{D} \sum_{j=1, j\neq i}^{D} \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2. \tag{8.4}$$

The motivation for using MPD is that it is more sensitive to multi-peak distributions than standard deviation, which is preferred when learning diverse content symbols in a sample.

Content variability within a sample ($\mathcal{V}_{\mathrm{f}}^{\mathrm{c}}$). We first compute the variability of content along the fragment axis and take the average along the sample axis. The value is the average of content codes before and after vector quantization:

$$\mathcal{V}_{\mathrm{f}}^{\mathrm{c}} = \frac{1}{2N} \sum_{i=1}^{N} \nu_{j=1}^{L}(\boldsymbol{z}_{ij}^{\mathrm{c}}) + \frac{1}{2N} \sum_{i=1}^{N} \nu_{j=1}^{L}(\tilde{\boldsymbol{z}}_{ij}^{\mathrm{c}}). \tag{8.5}$$

Content variability across samples ($\mathcal{V}_{\mathrm{s}}^{\mathrm{c}}$). Theoretically, we aim to measure the consistency of codebook usage distribution along the sample axis, which is not differentiable. In practice, we compute the center of the content code along the fragment axis and measure the variability of the centers along the sample axis. It serves as a proxy of codebook utilization. Also, we consider both content codes before and after vector quantization:

$$\mathcal{V}_{\mathrm{s}}^{\mathrm{c}} = \frac{1}{2}\nu_{i=1}^{N}\big(\frac{1}{L}\sum_{j=1}^{L} \boldsymbol{z}_{ij}^{\mathrm{c}}\big) + \frac{1}{2}\nu_{i=1}^{N}\big(\frac{1}{L}\sum_{j=1}^{L} \tilde{\boldsymbol{z}}_{ij}^{\mathrm{c}}\big). \tag{8.6}$$

Style variability within a sample ($\mathcal{V}_{\mathrm{f}}^{\mathrm{s}}$). We compute the variability of style representations among fragments and take their mean across all samples:

$$\mathcal{V}_{\mathrm{f}}^{\mathrm{s}} = \frac{1}{N} \sum_{i=1}^{N} \nu_{j=1}^{L}(\boldsymbol{z}_{ij}^{\mathrm{s}}). \tag{8.7}$$

Style variability across samples ($\mathcal{V}_{\mathrm{s}}^{\mathrm{s}}$). We compute the average style representation along the fragment axis and measure its variability along the sample axis:

$$\mathcal{V}_{\mathrm{s}}^{\mathrm{s}} = \nu_{i=1}^{N}\big(\frac{1}{L}\sum_{j=1}^{L}\boldsymbol{z}_{ij}^{\mathrm{s}}\big). \tag{8.8}$$

8.2.3 Variance-Versus-Invariance (V3) Constraints

With the variability statistics, we can formalize the general relationship between content and style along the sample or fragment axis:

- Content should be more variable within samples than across samples, i.e., $\mathcal{V}_{\mathrm{f}}^{\mathrm{c}} \gg \mathcal{V}_{\mathrm{s}}^{\mathrm{c}}$.
- Style should be more variable across samples than within samples, i.e., $\mathcal{V}_{\mathrm{s}}^{\mathrm{s}} \gg \mathcal{V}_{\mathrm{s}}^{\mathrm{f}}$.
- Within a sample, content should be more variable than style, i.e, $\mathcal{V}_{\mathrm{f}}^{\mathrm{c}} \gg \mathcal{V}_{\mathrm{f}}^{\mathrm{s}}$.
- Across samples, style should be more variable than content, i.e., $\mathcal{V}_{\mathrm{s}}^{\mathrm{s}} \gg \mathcal{V}_{\mathrm{s}}^{\mathrm{c}}$.

We quantify the above contrasts as regularization terms, using the hinge function to cut off gradient back-propagation when the ratio between two variability statistics reaches a certain threshold $r > 1$, which stands for relativity [11]:

$$\mathcal{L}_{\text{content}} = \max(0, 1 - \frac{\mathcal{V}_{\mathrm{f}}^{\mathrm{c}}}{r \cdot \mathcal{V}_{\mathrm{s}}^{\mathrm{c}}}), \quad (\mathcal{V}_{\mathrm{f}}^{\mathrm{c}} \gg \mathcal{V}_{\mathrm{s}}^{\mathrm{c}}) \tag{8.9}$$

$$\mathcal{L}_{\text{style}} = \max(0, 1 - \frac{\mathcal{V}_{\mathrm{s}}^{\mathrm{s}}}{r \cdot \mathcal{V}_{\mathrm{f}}^{\mathrm{s}}}), \quad (\mathcal{V}_{\mathrm{s}}^{\mathrm{s}} \gg \mathcal{V}_{\mathrm{f}}^{\mathrm{s}}) \tag{8.10}$$

$$\mathcal{L}_{\text{fragment}} = \max(0, 1 - \frac{\mathcal{V}_{\mathrm{f}}^{\mathrm{c}}}{r \cdot \mathcal{V}_{\mathrm{f}}^{\mathrm{s}}}), \quad (\mathcal{V}_{\mathrm{f}}^{\mathrm{c}} \gg \mathcal{V}_{\mathrm{f}}^{\mathrm{s}}) \tag{8.11}$$

$$\mathcal{L}_{\text{sample}} = \max(0, 1 - \frac{\mathcal{V}_{\mathrm{s}}^{\mathrm{s}}}{r \cdot \mathcal{V}_{\mathrm{s}}^{\mathrm{c}}}). \quad (\mathcal{V}_{\mathrm{s}}^{\mathrm{s}} \gg \mathcal{V}_{\mathrm{s}}^{\mathrm{c}}) \tag{8.12}$$

We obtain the V3 regularization term (used in Equation 8.1) by summing up the four terms:

$$\mathcal{L}_{\text{V3}} = \mathcal{L}_{\text{content}} + \mathcal{L}_{\text{style}} + \mathcal{L}_{\text{fragment}} + \mathcal{L}_{\text{sample}}. \tag{8.13}$$

8.3 Experiments

We evaluate V3 on both synthetic and real data to evaluate its effectiveness and generalizability in different domains and scenarios, covering audio, image, and video data. The highlight of this section is that V3 effectively learns disentangled representations of content and style, performs well on out-of-distribution generalization, and the discrete content representations manifest symbolic-level interpretability that aligns well with human knowledge.

We compare V3 with three unsupervised baselines: 1) an unsupervised content-style disentanglement based on MINE [232], and 2) a 2-branch autoencoder similar to our architecture choice, but trained with the cycle consistency loss after decoding and encoding shuffled combinations of $\tilde{\boldsymbol{z}}^{\mathrm{c}}$ and $\boldsymbol{z}^{\mathrm{s}}$ [290]. 3) a vanilla β-VAE [98]. Additionally, we compare with two methods with label supervision: 1) a weakly-supervised method for disentanglement named EC^2-VAE, in which the model is trained to predict the correct content labels from $\boldsymbol{z}^{\mathrm{c}}_{ij}$ as a replacement of the VQ layer, and the decoder is trained to reconstruct inputs from $\boldsymbol{z}^{\mathrm{s}}_{ij}$ and ground truth content labels [247, 271], and 2) a fully supervised variant of EC^2-VAE provided with both content and style labels, in which the model learns to predict both content and style from their latent representations. We denote them as EC^2-VAE (c) and EC^2-VAE (c & s) respectively. All reported results are the average of the three best-performing checkpoints on validation sets.

8.3.1 Datasets

Written Phone Numbers Dataset (PhoneNums): We synthesize an image dataset of written digit strings on light backgrounds using 8 different ink colors, mimicking a scenario of handwritten phone numbers. The order of digits is random. All images are diversified with noises, blur, and foreground and background color jitters. Models should learn digits and colors as content and style.

Monophonic Instrument Notes Dataset (InsNotes): We synthesize a dataset consisting of 16kHz monophonic music audio of 12 different instruments playing 12 different pitches in an octave. Every pitch is played for one second with a random velocity and amplitude envelope. The audio files are then normalized and processed to magnitude spectrograms. Models should learn pitches and timbres as content and style, respectively.

Street View House Numbers (SVHN) [182]: We select all images with more than one digit from the SVHN dataset. We crop the images to the bounding boxes of the digits and resize them to 32×48. Models should learn digits as content, their fonts, texture, and colors as style. Note that the styles can be seen as from a continuous space, and the fonts in SVHN are very diverse.

Sprites with Actions Dataset (Sprites) [147, 149]: The Sprites dataset contains animated cartoon characters with random appearances. We use a modified version of the dataset taking video sequences of characters performing 9 different actions in random order. Models should learn the actions as content and appearances as style. Note that the styles can be seen as from a continuous space.

8.3.2 Results of Content-Style Disentanglement

On PhoneNums and InsNotes where concrete style labels are available, we evaluate the models' content-style disentanglement ability by conducting a retrieval experiment to examine the nearest neighbors of every input $\boldsymbol{z}^{\mathrm{c}}$ and $\boldsymbol{z}^{\mathrm{s}}$ using ground truth content and style labels, evaluated by the area under the precision-recall curve (PR-AUC) and the best F1 score. We experiment with different codebook sizes K to allow different levels of vocabulary redundancy. The results are shown in Table 8.1 and Table 8.2. We see that V3 outperforms unsupervised baselines on both datasets, and the performance is consistent across different codebook sizes K. V3 also outperforms EC^2-VAE (c) in the style retrieval task, which indicates that V3 learns better-disentangled style representations containing less content information.

On SVHN and Sprites where there are no style labels, we evaluate the models' disentanglement ability by linear probing on the learned representations to predict content labels. We also

Method	K	Content				Style			
		PR-AUC		Best F1		PR-AUC		Best F1	
		$\boldsymbol{z}^{\mathrm{c}} \uparrow$	$\boldsymbol{z}^{\mathrm{s}} \downarrow$	$\boldsymbol{z}^{\mathrm{c}} \uparrow$	$\boldsymbol{z}^{\mathrm{s}} \downarrow$	$\boldsymbol{z}^{\mathrm{c}} \downarrow$	$\boldsymbol{z}^{\mathrm{s}} \uparrow$	$\boldsymbol{z}^{\mathrm{c}} \downarrow$	$\boldsymbol{z}^{\mathrm{s}} \uparrow$
V3	10	83.2	12.8	84.1	18.5	14.9	**95.4**	**22.6**	91.0
	20	**93.0**	11.6	**92.9**	18.2	**11.9**	93.9	22.7	89.9
	40	86.3	**10.9**	83.4	**18.0**	15.5	95.3	22.9	**93.0**
MINE-based	10	33.8	21.6	36.0	30.8	14.0	35.5	22.7	38.6
	20	41.9	25.0	49.5	25.4	22.2	37.5	33.2	39.0
	40	46.8	23.8	49.8	28.0	26.6	37.7	27.6	48.8
Cycle loss	10	55.1	25.4	64.3	27.8	17.0	33.1	**22.6**	37.4
	20	52.0	23.6	58.5	29.2	18.2	35.9	23.4	38.8
	40	53.8	20.7	62.8	22.4	19.3	31.6	24.5	35.8
β-VAE	-	24.9		27.0		25.6		30.5	
EC^2-VAE (c)	-	95.2	11.5	95.1	18.0	16.8	57.7	22.6	57.2
EC^2-VAE (c & s)	-	95.2	13.6	95.1	19.6	25.2	96.2	31.0	91.0

Table 8.1: Evaluation of digit and color disentanglement on PhoneNums using latent retrieval. Values are reported in percentage.

compare with a linear classifier on raw input features. The classifier layer is trained for one epoch before being evaluated on the test set. On both datasets, we allow a 100% content vocabulary redundancy, resulting in $K = 20$ for SVHN and $K = 18$ for Sprites. The resulting accuracies are shown in Table 8.3 and Table 8.4. V3 outperforms unsupervised baselines on both datasets, only trailing behind the weakly supervised EC^2-VAE (c) as the latter's $\boldsymbol{z}^{\mathrm{c}}$ space is optimized for the discriminative task.

8.3.3 Content Classification on Out-of-distribution Styles

We further evaluate the generalization ability of V3 on PhoneNums and InsNotes by testing the models' content classification performance on a special test set with only **unseen** styles, provided with few-shot examples. We focus on comparing V3 with the weakly supervised method EC^2-VAE (c) and a pure CNN classifier to evaluate the generalization ability introduced by latent

Method	K	Content				Style			
		PR-AUC		Best F1		PR-AUC		Best F1	
		$\boldsymbol{z}^{\mathrm{c}} \uparrow$	$\boldsymbol{z}^{\mathrm{s}} \downarrow$	$\boldsymbol{z}^{\mathrm{c}} \uparrow$	$\boldsymbol{z}^{\mathrm{s}} \downarrow$	$\boldsymbol{z}^{\mathrm{c}} \downarrow$	$\boldsymbol{z}^{\mathrm{s}} \uparrow$	$\boldsymbol{z}^{\mathrm{c}} \downarrow$	$\boldsymbol{z}^{\mathrm{s}} \uparrow$
V3	12	**89.9**	8.9	**90.1**	15.1	**9.3**	**87.5**	**15.0**	**88.0**
	24	76.2	8.7	80.0	14.2	12.8	68.9	20.3	70.0
	48	72.2	8.4	74.4	**14.2**	12.3	72.2	22.0	71.5
MINE-based	12	56.4	**7.61**	62.0	14.2	10.3	61.4	16.9	63.7
	24	50.5	8.5	59.1	14.9	14.7	53.4	19.5	51.4
	48	44.6	10.2	54.0	16.5	13.8	52.1	18.3	49.7
Cycle loss	12	49.7	8.7	57.9	15.2	10.7	12.7	18.2	19.0
	24	47.0	8.7	54.5	15.2	14.2	18.9	19.4	23.1
	48	42.4	8.0	49.4	14.5	16.2	20.0	22.4	24.4
β-VAE	-	18.1		20.8		12.2		19.0	
EC^2-VAE (c)	-	83.2	8.0	86.2	14.2	10.7	60.0	16.9	62.8
EC^2-VAE (c & s)	-	90.4	7.9	90.4	14.2	11.1	90.5	18.0	90.4

Table 8.2: Evaluation of pitch and timbre disentanglement on InsNotes using latent retrieval. Values are reported in percentage.

disentanglement. In the n-shot settings, models are presented with n samples of each content and new style combination. All models are continuously trained on new samples until performance stops improving. For V3, we choose the V3 versions with no codebook redundancy for comparison ($K = 10$ for PhoneNums, $K = 12$ for InsNotes) as they show a one-to-one mapping from codebook entries to content labels (see Section 8.3.4 for details). We first align the learned codebook entries to ground truth content labels, and obtain classification results by the encoded content representations $\boldsymbol{z}^{\mathrm{c}}$. For EC^2-VAE, we try two different continuous training strategies: 1) using pseudo content labels from its own predictions for self-boosting, as well as training the reconstruction loss, and 2) only optimizing the reconstruction loss. Additionally, we compare with EC^2-VAE and the CNN classifier provided with labels in continuous training. The results are shown in Table 8.5. Although V3 might fall behind supervised methods in the 0-shot setting, it comes to the lead in few-shot settings on both datasets as the number of extra samples increases.

Method	K	$z^c \uparrow$	$z^s \downarrow$
V3	20	**40.6**	**18.5**
MINE-based	20	36.0	20.8
Cycle loss	20	16.8	21.2
β-VAE	-	21.8	
Raw input	-	21.4	
EC^2-VAE (c)	-	97.0	21.2

Table 8.3: Linear probing accuracies (in %) for content (digit) classification on SVHN.

Method	K	$z^c \uparrow$	$z^s \downarrow$
V3	18	**88.2**	**20.2**
MINE-based	18	79.1	22.2
Cycle loss	18	86.4	39.7
β-VAE	-	33.2	
Raw input	-	99.0	
EC^2-VAE (c)	-	99.8	15.7

Table 8.4: Linear probing accuracies (in %) for content (action) classification on Sprites.

This indicates that V3 can learn by itself to make sense of unseen styles with only a few examples, an ability that emerges from learning representations and disentangled interpretable factors.

Pretraining		Continuous Training		PhoneNums				InsNotes			
Method	Supervision	Supervision	Self-boost	0-shot	1-shot	5-shot	10-shot	0-shot	1-shot	5-shot	10-shot
V3	No	No	No	57.8	91.3	**97.1**	**99.0**	90.5	**97.6**	**97.8**	**99.2**
EC^2-VAE (c)	Yes	No	No	**84.2**	**92.1**	92.2	92.7	87.1	87.2	89.4	91.2
EC^2-VAE (c)	Yes	No	Yes	**84.2**	91.8	92.1	92.4	87.1	94.6	95.0	95.1
CNN Classifier	Yes	No	No	59.5	59.5	59.5	59.5	**92.6**	92.6	92.6	92.6
CNN Classifier	Yes	No	Yes	59.5	80.2	82.2	82.7	**92.6**	87.6	85.9	85.3
EC^2-VAE (c)	Yes	Yes	No	84.2	94.6	98.8	99.2	87.1	97.7	98.9	99.8
CNN Classifier	Yes	Yes	No	59.5	81.2	82.4	83.5	92.6	91.9	91.3	89.1

Table 8.5: Content classification accuracies (in %) on data with OOD styles.

8.3.4 Results of Symbolic Content Interpretability

We notice that interpretable symbols emerge in the learned codebook of V3, showing its ability of abstracting concepts from information. To evaluate the interpretability of learned content representations quantitatively, we propose two metrics: the learned content codebook accuracy and standard deviation among styles. We first align codebook entries to the ground truth content labels by their distributions on content labels, and then calculate the accuracy of codebook entries' distribution regarding their aligned labels. A well learned, interpretable codebook should have entries concentrated on content labels they are aligned with, thus showing high accuracy.

Method	PhoneNums			InsNotes		
	K	Acc. ↑	σ ↓	K	Acc. ↑	σ ↓
V3	10	**89.2**	**0.6**	12	**99.8**	**0.1**
	20	**99.7**	**1.8**	24	**92.9**	**2.2**
	40	**99.9**	4.5	48	**90.2**	4.5
MINE-based	10	40.9	8.1	12	13.8	3.8
	20	25.6	9.8	24	29.4	8.9
	40	50.6	6.2	48	26.9	**3.6**
Cycle loss	10	71.0	3.7	12	27.5	11.4
	20	89.6	4.3	24	28.5	11.9
	40	**99.9**	**4.1**	48	18.2	6.2

Table 8.6: Quantitative results of codebook interpretability on datasets with discrete style labels. Values are reported in percentage.

Also, as a good symbol is a symbol of consensus, on datasets with discrete style labels, we also quantify different styles' discrepancy of codebook entries distribution on content labels using the standard deviation (σ) of confusion matrices between codebook entries and content labels, as shown in Table 8.6. For datasets with no discrete style labels, we report the accuracy of codebook entries in Table 8.7.

From Table 8.6 and Table 8.7, we observe that V3 shows good codebook interpretability by representing content labels with consistent codebook entries, and the consistency is kept well among styles. Table 8.6 also shows that V3 shows good codebook interpretability with or without vocabulary redundancy, indicating V3 does not rely on the knowledge of the number of content classes to learn interpretable symbols.

Qualitatively, we perform content and style recombination by traversing all content codebook entries and decoding them with a fixed style representation. If the learned codebook has good interpretability, the decoding results of recombined content and style representations should show meaningful content changes, and retain consistent styles. Here, we focus on comparing the recombination results of V3 and baselines on SVHN, where we first encode $\boldsymbol{z}^{\mathrm{s}}$ from example fragments, and then recombine it with all K content codebook entries.

Method	SVHN		Sprites	
	K	Acc. ↑	K	Acc. ↑
V3	20	**47.6**	18	**98.5**
MINE-based	20	26.0	18	38.3
Cycle loss	20	20.1	18	82.0

Table 8.7: Quantitative results of codebook interpretability on datasets without style labels. Values are reported in percentage.

From Figure 8.3, we observe that V3 generates images with clear content changes and consistent styles when recombining content and style representations. Although images generated by V3 may not cover all possible contents, as many of them never appear in the training set in the given styles, the contents are almost all recognizable digits, and V3 can even "imagine" reasonably what a digit would look like in a new font and color. In contrast, the baselines generate images with either mixed content and style information or very subtle changes in content that are hard to interpret. This comparison not only validates the interpretability of V3's learned codebook resulting from successful content-style disentanglement but also demonstrates the potential of V3 in style transfer and content editing tasks.

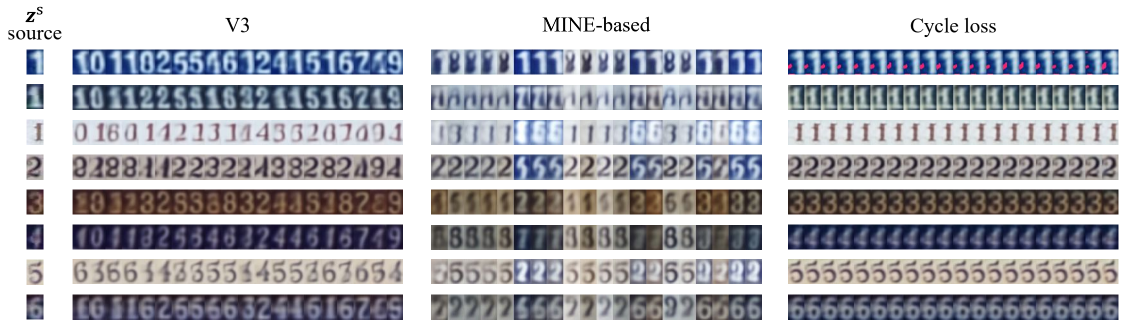


Figure 8.3: Comparison of generated images by recombining $\boldsymbol{z}^{\mathrm{s}}$ from given sources in SVHN and all $\boldsymbol{z}^{\mathrm{c}}$ in the learned codebook.

8.4 Related Work

The content-style disentanglement, as well as the related style transfer problem, has been well explored in computer vision, especially in the context of image-to-image translation. Early works mostly require paired data of the same content with different styles [111, 206], until the introduction of domain transfer networks that can learn style transfer functions without paired data [24, 44, 45, 120, 153, 191, 221, 263, 264, 290]. Although these methods are unsupervised in the sense that they do not require paired data, they still require concrete labels of styles to identify source and target domains, and there are no fully interpretable representations of either content or style.

A similar trajectory of research has also been followed in other domains, including speech [116, 117, 197, 261] and music [20, 150, 152, 159, 163, 284]. To mitigate the requirement for supervision, some methods utilize domain-specific knowledge and have achieved better disentanglement results, including X-vectors of speakers [197, 198], the close relation between fundamental frequency and content in audio [198, 199], or pre-defined style or content representations [245, 247, 271].

Pure unsupervised learning for content and style disentanglement has not been well explored. Notable attempts include mutual information-based methods such as InfoGAN and mutual information neural estimation (MINE) [15, 39, 196, 232, 280], and low-dimensional representation learning with physical symmetry [155]. But these methods often suffer from the training stability issue or have to follow a low-dimensionality setup. Disentangled Sequential Autoencoder (DSAE) and its variants leverage the nature of content and style to learn their representations at different scales, but their applications are limited to purely sequential data with a static style [9, 103, 147, 163, 164].

A technique often associated with learned content is vector quantization (VQ) [184]. Recent efforts have built language models on top of VQ codes for long-term generation, indicating the association between VQ codebook and the underlying information content [52, 75, 226, 231, 232, 239, 268]. A noticeable characteristic of these studies is the use of large codebooks, which limits the interpretability of representations. We borrow the idea of a *small* codebook size from categorical representations [39, 113], targeting a more concise and unified content code across different styles, while keeping the high-dimensional nature of VQ representations.

8.5 Limitation

We have identified several limitations in our V3 method that necessitate further investigation. First, while V3 achieves good disentanglement and symbolic interpretability, it is not flawless—samples of different contents (say images of "8" and "9") may be projected into the same latent code. Inspired by human learning, which effectively integrates both mode-1 and mode-2 cognitive processes, we aim to enhance V3 by incorporating certain feedback or reinforcement. This adaptation could also facilitate the application of V3 to more complex domains, such as general image or video. Additionally, V3 is currently optimized to disentangle content and style from data samples that include defined fragments. Extending this capability to unsegmented data of large vocabularies, such as continuous audio, represents a significant area for future development. Furthermore, V3 assumes that content elements do not overlap, which does not hold in cases of polyphonic music or mixed audio. Addressing this challenge will require a more sophisticated approach that considers the hierarchical nature of content.

8.6 Summary

In summary, we contributed an unsupervised content-style disentanglement method named V3. V3's inductive bias is domain-general, intuitive, and concise, solely based on the meta-level insight of the statistical difference between content and style, i.e., their distinct variance-invariance patterns reflected both within and across data samples. Experiment results showed that V3 not only outperforms the baselines in terms of content-style disentanglement but also demonstrates superior generalizability on OOD styles compared to supervised methods, and achieves high interpretability of learned content symbols. The effectiveness of V3 generalizes across different domains, including audio, image, and video. We believe that V3 has the potential to be applied to emergent knowledge in general, and we plan to extend our method to more complex tasks and domains in the future.

9 | Conclusion

This dissertation set out to explore a fundamental question: Can machines possess musicianship, so that they can engage with us meaningfully in the creative process? To answer this question, we analyzed the role of concept manipulation in music creation and introduced the framework of concept alignment: the idea that human-like musicianship in AI emerges from aligning the model's internal representations with music concepts through specialized learning and generation methods.

Across music generation tasks, we explored how AI systems can achieve this alignment. In concept representation, we developed methods to learn interpretable and disentangled representations of music factors, such as pitch contour, rhythm, chord progression, and texture, supporting nuanced control and style transfer. In concept organization, we introduced hierarchical generative models that capture both the compositional hierarchy of musical structure and the concept hierarchy of music ideas, enabling structured long-term generation. In concept emergence, we proposed domain-general inductive biases that allow the unsupervised discovery of symbolic abstractions, such as pitch and timbre or digit and color, across different modalities. These contributions demonstrate that concept alignment can improve controllability and interpretability in various generation tasks.

Over the years in which this thesis was developed, generative AI has advanced at an extraordinary pace. Large language models (LLMs), diffusion architectures, self-supervised learning, and new techniques for model grounding and multimodal understanding have reshaped the landscape of machine learning, making generative systems more capable than ever. As these capabilities expand, so too does the need for models that act meaningfully and align more closely with human creative intent. Concept alignment offers a path to bridge this gap.

Looking ahead, I see two main directions for advancing concept alignment. First, we can investigate how aligned methods can incorporate the most powerful generative tools, combining their expressive capacity with stronger grounding on concepts. This includes both implicit alignment—aligning the dynamics of models across different modalities without fully dissecting their internal mechanisms (discussed in Section 9.1); and explicit alignment—ensuring that concepts emerging within the model can be matched to and interpreted in human terms (discussed in Section 9.2). Second, with these methods in place, we can focus on applications in human-computer interaction, designing user interfaces that bring these models into the creative process in ways that genuinely enhance human-AI co-creation (discussed in Section 9.3).

9.1 Implicit Alignment of Multimodal Dynamics

Current state-of-the-art generative models are often monolithic: breaking them into fully interpretable components is difficult, let alone disentangling them into multiple concepts. Nevertheless, experiments suggest that many such concepts are already embedded within these models and can be revealed or leveraged through methods such as probing, in-context learning, and parameter-efficient fine-tuning. Building on this observation, the future direction of implicit alignment treats models as given—assuming their internal representations and concept manipulation abilities already exist—and focuses on aligning their capabilities across modalities, thereby making concept understanding more reliable and mutually reinforcing. Existing multimodal fine-tuning can be viewed as a step in this direction, but it should do more than just use one modality's embeddings to support another.

The most notable aspect of the proposal is not aligning embeddings, but rather the *generation* *dynamics*. For example, if an audio model and a symbolic music model were aligned in dynamics, could the symbolic model offer richer theoretical or structural insights to guide audio prediction? Conversely, could the audio model inspire more imaginative or stylistically nuanced symbolic generation? Similarly, if a text LLM and a music LLM were aligned in dynamics, could concepts of music (captured by the music LLM) and concepts about music (captured by the text LLM) be shared and mutually reinforced? Such alignment could open the door to richer reasoning, even back in the single modality, as the grounding of concepts is reinforced by other modalities.

9.2 Explicit Alignment in Cross-Domain Emergence

Despite the potential of implicit alignment, a fundamental question about intelligence remains: how do abstract concepts, categories, and hierarchies emerge, and how do they evolve and fade over time? Building on this thesis's findings on concept emergence, future work will investigate how symbols and higher-level concepts arise from raw observations. Examples range from the emergence of notes from audio, text from speech, or objects from video, to more complex semantics such as call-and-response patterns between musical phrases, narrative arcs in writing, or shot transitions in film.

These processes, though different, appear to be driven by unified, domain-general inductive biases, which I hypothesize give rise to a *hierarchical disentanglement of content and style* in a self-supervised manner. Formalizing these biases could enable models to acquire more reliable and generalizable planning abilities, discover new concepts in data-sparse settings, and adapt to unfamiliar domains. Ultimately, these methods could be integrated with implicit alignment, together yielding models that are more capable and interpretable.

9.3 Human-AI Co-Creation

Lastly, concept alignment opens new opportunities for human-AI co-creation, and its real value will only be revealed through direct exploration in human-computer interaction. A key question is whether the compositional and concept hierarchies studied in this thesis not only serve as effective representations for machines, but are also intuitive for human musicians to interact with. AI-assisted composition tools provide a natural testbed for this investigation, particularly by integrating hierarchical models into interactive music generation interfaces, such as digital audio workstations. In such environments, users could engage with AI at multiple levels: providing examples, shaping musical ideas, controlling global structure, and even observing how the system reasons about its choices.

Beyond composition tools, concept-aligned systems could extend to a variety of impactful applications. For example, they could enable interactive educational platforms that learn intuitive, example-based concepts and express them in multiple styles; create embodied AI performers capable of meaningful, real-time exchange with human musicians; or support music therapy where musical parameters are precisely controlled for targeted outcomes. In all of these scenarios, concept alignment serves as a foundation for designing systems whose reasoning and output are transparent, responsive, and capable of collaborating seamlessly with humans.

In summary, this dissertation proposed concept alignment and its future directions as a step toward musicianship in AI. May it serve as a reminder that aesthetics and art are part of intelligence, and as an invitation to explore how computational methods can find their path in harmony with humanity.

Appendices

A | PianoTree VAE for Polyphonic Representation Learning

This chapter is based on the published work *PianoTree VAE: Structured Representation Learning for Polyphonic Music* [248], conducted in collaboration with Yiyi Zhang, Yixiao Zhang, Junyan Jiang, Ruihan Yang, Junbo Zhao, and Gus Xia. The proposed model serves as the sub-module of the models proposed in Chapter 3 and Chapter 4.

The dominant approach for music representation learning involves the deep unsupervised model family *variational autoencoder* (VAE). However, most, if not all, viable attempts on this problem have largely been limited to monophonic music. Normally composed of richer modality and more complex musical structures, the polyphonic counterpart has yet to be addressed in the context of music representation learning. In this work, we propose the PianoTree VAE, a novel tree-structure extension upon VAE aiming to fit the polyphonic music learning.

The experiments prove the validity of the PianoTree VAE via (i)-semantically meaningful latent code for polyphonic segments; (ii)-more satisfiable reconstruction aside of decent geometry learned in the latent space; (iii)-this model's benefits to the variety of the downstream music generation.[1]

[1]Code and demos can be accessed via `https://github.com/ZZWaang/PianoTree-VAE`

A.1 Introduction

Unsupervised machine learning has led to a marriage of symbolic learning and vectorized representation learning [25, 47, 288]. In the computer music community, the MusicVAE [203] enables the interpolation in the learned latent space to render smooth music transitions. The EC^2-VAE [271] manages to disentangle certain interpretable factors in music and also provides a manipulable generation pathway based on these factors. Pati *et al.* [193] further utilizes the recurrent networks to learned music representations for longer-term coherence.

Unfortunately, most of the success has been limited to monophonic music. The generalization of the learning frameworks to polyphonic music is not trivial, due to its much higher dimensionality and more complicated musical syntax. The commonly-adopted MIDI-like event sequence modeling or the piano-roll formats fed to either recurrent or convolutional networks have fell short in learning good representation, which usually leads to unsatisfied generation results [65, 215, 270]. In this chapter, we hope to pioneer the development of this challenging task. To begin with, we conjecture a proper set of **inductive bias** for the desired framework: (i)-a sparse encoding of music as the model input; (ii)-a neural architecture that incorporates the hierarchical structure of polyphonic music (i.e., musical syntax).

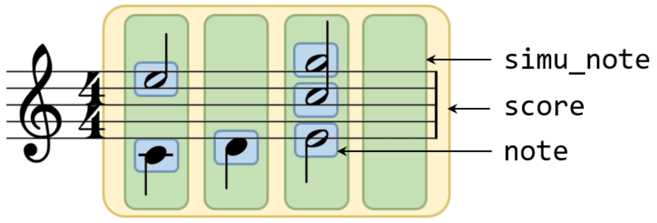


Figure A.1: An illustration of the proposed polyphonic syntax.

Guided by the aforementioned design principles, we propose PianoTree VAE, a hierarchical representation learning model under the VAE framework. We adopt a tree structured musical syntax that reflects the hierarchy of musical concepts, which is shown in Figure A.1. In a top-down order: we define a `score` (indicated by the yellow rectangle) as a series of `simu_note` events (indicated by the green rectangles), a `simu_note` as multiple `note` events sharing the same onset (indicated by blue rectangles), and each `note` has several attributes such as *pitch* and *duration*. In this chapter, we focus on a simple yet common form of polyphonic music—piano score, in which each note has only pitch and duration attributes. For future work, this syntax can be generalized to multiple instruments and expressive performance by adding extra attributes such as voice, expressive timing, dynamics, etc.

The whole neural architecture of PianoTree VAE can be seen as a tree. Each node represents the embedding of either a `score`, `simu_note`, or `note`, where a higher level representation has larger receptive fields. The edges are bidirectional, where a recurrent module is applied to either encode the children into the parent or decode the parent to generate its children.

Through extensive evaluations, we show that PianoTree VAE yields semantically more meaningful latent representations and further downstream generation quality gains, on top of the current state-of-the-art solutions.

A.2 Related Work

The complex hierarchical nature of music data has been studied for nearly a century (e.g., GTTM [141], Schenkerian Analysis [204], and their follow-up works [87, 89, 170, 217]). However, the emerging deep representation-learning models still lack compatible solutions to deal with the complex musical structure. In this section, we first review different types of polyphonic music generation in Section A.2.1. After that, we discuss some popular deep music generative models indexed by their compatible data structure from Section A.2.2 to Section A.2.4.

A.2.1 Different Types of Polyphony

In the context of deep music generation, *polyphony* can refer to three types of music: 1) multiple monophonic parts (e.g., a four-part chorus), 2) a single part of a polyphonic instrument (e.g., a piano sonata), and 3) multiple parts of polyphonic instruments (e.g., a symphony).

The first type of polyphonic music can be created by simply extending the number of voices in monophonic music generation with some inter-voice constraints. Some representative systems belonging to this category include DeepBach [83], XiaoIce [289], and Coconet [105]. Music Transformer [106] and the proposed PianoTree VAE both focus on the generation of the second type of polyphony, which is a much more difficult task. Polyphonic pieces under the second definition no longer have a fixed number of "voices" and consist of more complex textures. The third type of polyphony can be regarded as an extension of the second type, and we leave it for future work.

A.2.2 Piano-roll and Compatible Models

Piano-roll and its variations [29, 65, 128, 169] view polyphonic music as 3-D (one-hot) tensors, in which the first two dimensions denote time and pitch, and the third dimension indicates whether the token is an onset, sustain, or rest. A common way for deep learning models to encode/decode a piano-roll is to use recurrent layers along the time-axis while the pitch-axis relations are modeled in various ways [23, 29, 169]. Another method is to regard a piano-roll as an image with three channels (onset, sustain, and rest) and apply convolutional layers [65, 128].

Through the proposal of PianoTree VAE, we argue that a major way to improve the current deep learning models is to utilize the built-in priors (intrinsic structure) in the musical data. In our work, we primarily use the sparsity and the hierarchical priors.

A.2.3 MIDI-like Event Sequence and Compatible Models

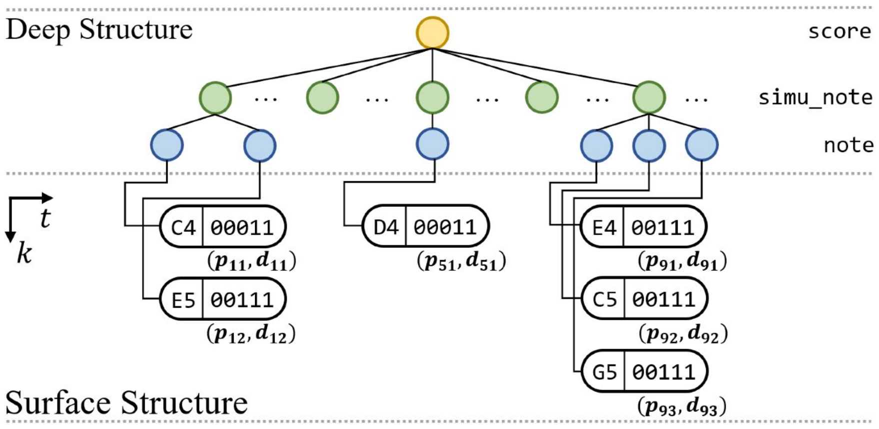
MIDI-like event sequence is first used in deep music generation in performanceRNN [214] and multi-track MusicVAE [215], and then broadly applied in transformer-based generation [64, 106, 108]. This direction of work leverages the sparsity of polyphonic data to efficiently flatten polyphonic music into an array of events. The vocabulary size of events usually triples the vocabulary size of MIDI pitches, including "note-on" and "note-off" events for 128 MIDI pitches, "time shifts", and so on.

However, the format of MIDI-like events lacks the proper flexibility. A few operations are made difficult due to its very nature. For instance, during addition or deletion of notes, often numerous "time shift" tokens must be merged or split with the "note-on" or "note-off" tokens being changed all-together. This has caused the model being trained inefficient for the potential generation tasks. In addition, this format has a risk of generating illegal sequences, say a "note on" message without a paired "note off" message.

Similarly, we see the note-based approaches [92, 176], in which polyphonic music is represented as a sequence of note tuples, as an alternative to the MIDI-like methods. The representation has resolved the illegal generation problem but still not revealed much of the intrinsic music structure. We argue that our work improves on the note-based approaches by utilizing deeper musical structures implied by the data. (See Section 3.1 for details.)

A.2.4 GNN as a Novel Structure

Recently, we see a trend in using graph neural networks (GNN) [207] to represent polyphonic score [112], in which each vertex represents a note and the edges represent different musical relations. Although the GNN-based model offers sparse representation learning capacity, it is limited by the specification of the graph structure design, and it is nontrivial to generalize it for score generations.



Deep Structure
score
simu_note
note
t
k
C4 00011
$(\boldsymbol{p}_{11}, \boldsymbol{d}_{11})$
E5 00111
$(\boldsymbol{p}_{12}, \boldsymbol{d}_{12})$
D4 00011
$(\boldsymbol{p}_{51}, \boldsymbol{d}_{51})$
E4 00111
$(\boldsymbol{p}_{91}, \boldsymbol{d}_{91})$
C5 00111
$(\boldsymbol{p}_{92}, \boldsymbol{d}_{92})$
G5 00111
$(\boldsymbol{p}_{93}, \boldsymbol{d}_{93})$
Surface Structure

Figure A.2: An illustration of PianoTree data structure to encode the music example in Figure A.1.

A.3 Method

A.3.1 Data Structure

We first define a data structure to represent a polyphonic music segment, which contains two components: 1) *surface structure*, a data format to represent the music observation, and 2) *deep structure*, a tree structure (containing `score`, `simu_note` and `note` nodes) showing the syntactic construct of the music segment.

Each music segment lasts T time steps with $\frac{1}{4}$ beat as the shortest unit. We further use K_t, where $1 \leq t \leq T$, to denote the number of notes having the same onset t. The current model uses $T = 32$, i.e., each music segment is 8-beat long.

Surface Structure

The surface structure is a nested array of *pitch-duration* tuples, denoted by $\{(p_{t,k}, d_{t,k}) | 1 \leq t \leq T, 1 \leq k \leq K_t\}$. Here, $(p_{t,k}, d_{t,k})$ is the k^{th} lowest note starting at time step t. The pitch attribute $p_{t,k}$ is a 128-D one-hot vector corresponding to 128 MIDI pitches. The duration attribute $d_{t,k}$ encodes the duration ranging from 1 to T using a $\log_2 T$-bit binary vector. For example, when $T = 32$ ($\log_2 T = 5$), '00000' represents a 16^{th} note, '00001' is an 8^{th} note, '00010' is a dotted 8^{th}

note, and so on so forth. The base-2 design is inspired by the similar binary relation among different note values in western musical notation.

The bottom part of Figure A.2 illustrates the surface structure of the music example in Figure A.1. We see that the data structure is a sparse encoding of music, and it eliminates illegal tokens since every possible nested array has a corresponding music.

Deep Structure

We further build a syntax tree to reveal the hierarchical relation of the observation. First, for $1 \leq t \leq T, 1 \leq k \leq K_t$, we define $\texttt{note}_{t,k}$ as the summary (i.e., embedding) of $(p_{t,k}, d_{t,k})$, which are the bottom layers of the tree. Then, for $1 \leq t \leq T$, we define $\texttt{simu\_note}_t$ as the summary of $\texttt{note}_{t,1\leq k \leq K_t}$, which are the middle layers of the tree. Finally, we define the `score` as the summary of $\texttt{simu\_note}_{1\leq t \leq T}$, which is the root of the tree. The upper part of Figure A.2 illustrates the deep structure built upon its surface structure.

The syntax tree, so-called the deep structure, has both musical and linguistic consideration. In terms of music, `note`, `simu_note`, and `score` roughly reflect the musical concept of a note, chord, and grouping. In terms of linguistics, the tree is analogous to a constituency tree, with surface structure being the terminal nodes and deep structure being the non-terminals. Recent studies in natural language processing have revealed that incorporating natural language syntax results in better semantics modeling [66, 220].

A.3.2 Model Structure

We use the surface structure of polyphonic music as the model input. The VAE architecture is built upon a structure.

We denote the music segment in the proposed surface structure as x and the latent code as z, which conforms to a standard Gaussian prior denoted by $p(z)$. The encoder models the approximated posterior $q_\phi(z|x)$ in a bottom-up order of the deep structure. First, `note` embeddings

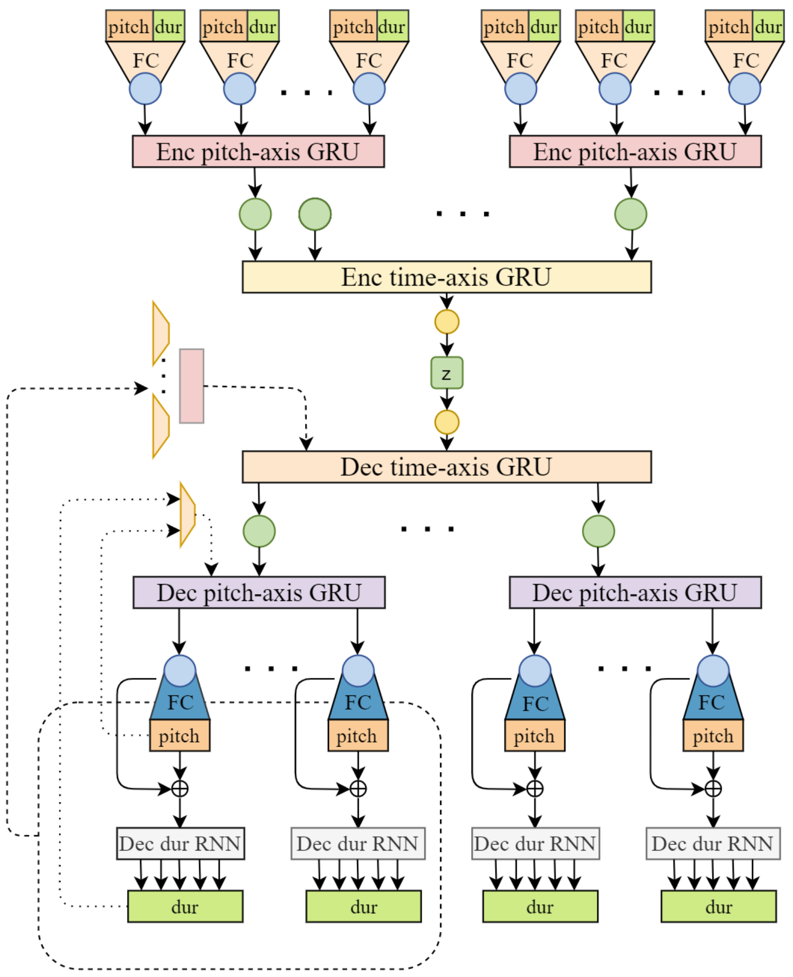


Figure A.3: An overview of the model architecture. The recurrent layers are represented by rectangles and the fully-connected (FC) layers are represented by trapezoids. The `note`, `simu_note`, and `score` events are represented by circles.

are computed through a linear transform of pitch-duration tuples. Second, the `note` embeddings (sorted by pitch) are then embedded into `simu_note` using a bi-directional GRU [41] by concatenating the last hidden states on both ends. With the same method, the `simu_note` embeddings (sorted by onsets) are summarized into `score` by another bi-directional GRU. We assume an isotropic Gaussian posterior, whose mean and log standard deviation are computed by a linear mapping of `score`. Algorithm 2 shows the details.

The decoder models $p_\theta(x|z)$ in a top-down order of the deep structure, almost mirroring the encoding process. We use a uni-directional time-axis GRU to decode `simu_note`, another uni-

Algorithm 2 The **PianoTree Encoder**. n, sn, sc are short for note, simu_note, score.

```
Input: PianoTree $x = \{(p_{t,k}, d_{t,k}) \mid 1 \le t \le T,\ 1 \le k \le K_t\}$
1: /* gru(·): passes a sequence to bi-directional GRU and outputs the
   concatenation of hidden states from both ends. */
2: for all $t, k$ do
3:     $\mathrm{n}_{t,k} \leftarrow \mathrm{emb}_{\mathrm{enc}}(p_{t,k}, d_{t,k})$
4: end for
5: for all $t$ do
6:     $\mathrm{sn}_t \leftarrow \mathrm{gru}^{\mathrm{pitch}}_{\mathrm{enc}}(\mathrm{n}_{t,1:K_t})$
7: end for
8: $\mathrm{sc} \leftarrow \mathrm{gru}^{\mathrm{time}}_{\mathrm{enc}}(\mathrm{sn}_{1:T})$
9: $\mu \leftarrow \mathrm{fc}_\mu(\mathrm{sc})$; $\sigma \leftarrow \exp(\mathrm{fc}_\sigma(\mathrm{sc}))$
10: return $q(z|x) = \mathcal{N}(\mu, \sigma^2)$
```

directional (pitch-axis) GRU to decode note, a fully connected layer to decode pitch attributes, and finally another GRU to decode duration attribute starting from the most significant bit. Algorithm 3 shows the details.

We use the ELBO (evidence lower bound) [125] as our training objective. Formally,

$$\mathcal{L}(\phi, \theta; x) = -\mathbb{E}_{z \sim q_\phi} \log p_\theta(x|z) + \beta \mathrm{KL}\Big(q_\phi || p(z)\Big), \tag{A.1}$$

where β is a balancing parameter used in β-VAE [98].

We denote the embedding size of note, simu_note and score as e_{n}, e_{sn} and e_{sc}; the dimension of latent space as d_{z}; and the hidden dimensions or pitch-axis, time-axis and dur GRUs as h_{p}, h_{t} and h_{d} respectively. In this work, we report our result on the following model size: $e_{\mathrm{n}} = 128$, $e_{\mathrm{sn}} = h_{\mathrm{p,dec}} = 2 \times h_{\mathrm{p,enc}} = 512$, $e_{\mathrm{sc}} = h_{\mathrm{t,dec}} = 2 \times h_{\mathrm{t,enc}} = 1024$, $h_{\mathrm{d,dec}} = 64$ and $d_z = 512$.

A.4 Experiments

In this section, we compare PianoTree VAE with several baseline models. We present the dataset in Section A.4.1, baseline models in Section A.4.2,and the training details in Section A.4.3. We present the objective evaluation on reconstruction accuracy in Section A.4.4. In Section A.4.5,

Algorithm 3 The **PianoTree Decoder**. We still use the abbreviation n, sn, and sc, defined in Algorithm 2.

Input: latent representation z

```
1: /* gru(·), same as Algorithm 1. */
2: /* grucell(·, ·): updates the hidden state using the current input and the
   previous hidden state. The output is replicated. */
3: $\mathtt{sc} \leftarrow z$
4: $\tilde{\mathtt{sn}}_0,\ \tilde{\mathrm{n}}_{:,0},\ d_{:,:,0} \leftarrow$ <SOS>
5: for $t = 1$ to $T$ do
6:     $[\mathtt{sn}_t,\ \mathtt{sc}] \leftarrow \mathrm{grucell}_{\mathrm{dec}}^{\mathrm{time}}(\tilde{\mathtt{sn}}_{t-1},\ \mathtt{sc})$
7:     for $k = 1$ to max pitch steps do
8:         $[\mathrm{n}_{t,k},\ \mathtt{sn}_t] \leftarrow \mathrm{grucell}_{\mathrm{dec}}^{\mathrm{pitch}}(\tilde{\mathrm{n}}_{t,k-1},\ \mathtt{sn}_t)$
9:         $p_{t,k} \leftarrow \mathrm{softmax}(\mathrm{fc}(\mathrm{n}_{t,k}))$
10:        for $r = 1$ to $5$ do
11:            $h \leftarrow [\mathrm{n}_{t,k},\ p_{t,k}]$
12:            $[y_{t,k,r},\ h] \leftarrow \mathrm{grucell}_{\mathrm{dec}}^{\mathrm{dur}}(d_{t,k,r-1},\ h)$
13:            $d_{t,k,r} \leftarrow \mathrm{softmax}(y_{t,k,r})$
14:        end for
15:        $d_{t,k} \leftarrow [d_{t,k,1:5}]$
16:        if $p_{t,k} \neq$ <EOS> then
17:            $K_t \leftarrow k$
18:            break
19:        end if
20:        $\tilde{\mathrm{n}}_{t,k} \leftarrow \mathrm{emb}_{\mathrm{enc}}(p_{t,k},\ d_{t,k})$
21:    end for
22:    $\tilde{\mathtt{sn}}_t \leftarrow \mathrm{gru}_{\mathrm{enc}}^{\mathrm{pitch}}(\mathrm{n}_{t,1:K_t})$
23: end for
24: return $\{(p_{t,k},\ d_{t,k}) \mid 1 \le t \le T,\ 1 \le k \le K_t\}$
```

we inspect and visualize the latent space of `note` and `simu_note`. After that, we present the subjective evaluation on latent space traversal in Section A.4.6. Finally, we apply the learned representation to downstream music generation task in Section A.4.7.

A.4.1 Dataset

We collect around 5K classical and popular piano pieces from Musicalion[2] and the POP909 dataset [250]. We only keep the pieces with $\frac{2}{4}$ and $\frac{4}{4}$ meters and cut them into 8-beat music

[2]Musicalion: https://www.musicalion.com.

segments (i.e., each data sample in our experiment contains 32 time steps under sixteenth note resolution). In all, we have 19.8K samples. We randomly split the dataset (at song-level) into the training set (90%) and the test set (10%). All training samples are further augmented by transposing to all 12 keys.

A.4.2 Baseline Model Architectures

We train four types of baseline models in total using piano-roll (Section A.2.2) and MIDI-like events (Section A.2.3) data structures. As a piano-roll can be regarded as either a sequence or a 2-dimensional image, we couple it with three neural encoder-decoder architectures: a recurrent VAE (**pr-rnn**), a convolutional VAE (**pr-cnn**), and a fully-connected VAE (**pr-fc**). For the MIDI-like events, we couple it with a recurrent VAE model (**midi-seq**). All models share the same latent space dimension ($d_z = 512$). Specifically,

- The piano-roll recurrent VAE (**pr-rnn**) model is similar to a 2-bar MusicVAE proposed in [203]. The hidden dimensions of the GRU encoder and decoder are both 1024.
- The piano-roll convolutional VAE (**pr-cnn**) architecture adopts a convolutional–deconvolutional architecture. The encoder contains 8 convolutional layers with kernel size 3×3. Strided convolution is performed at the 3rd, 5th, 7th and 8th layer with stride size $(2 \times 1), (2 \times 3), (2 \times 2)$ and (2×2) respectively. The decoder adopts the deconvolution operations in a reversed order.
- The piano-roll fully-connected VAE (**pr-fc**) architecture uses a time-distributed 256-dimensional embedding layer, followed by 3 fully-connected layers with the hidden dimensions [1024, 768] for the encoder. The decoder adopts the counter-operations in a reversed order.
- The MIDI-like event recurrent VAE (**midi-seq**) adopts the recurrent model structure similar

to **pr-rnn**. Here, the event vocabulary contains 128 "note-on", 128 "note-off" and 32 "time shift" tokens. The embedding size of a single MIDI event is 128. The hidden dimensions of the encoder GRU and decoder GRU are 512 and 1024 respectively.

A.4.3 Training

For all models, we set batch size $= 128$ and use Adam optimizer [124] with a learning rate starting from 1e-3 with exponential decay to 1e-5. For PianoTree VAE, we use teacher forcing [234] for the decoder time-axis and pitch-axis GRU, and for other recurrent-based baselines, we use teacher forcing in the decoders. The teacher forcing rates start from 0.8 and decay to 0.0. PianoTree VAE converges within 6 epochs, and the baseline models converge in approximately 40 to 60 epochs.

Models	**PianoTree**	**midi-seq**	**pr-rnn**	**pr-cnn**	**pr-fc**
Onset Precision	**0.9558**	0.8929	0.9533	0.9386	0.9211
Onset Recall	**0.9532**	0.6883	0.9270	0.8818	0.8827
Onset F1	**0.9545**	0.7774	0.9399	0.9093	0.9015
Duration Precision	**0.9908**	0.3826	0.9777	0.9757	0.9688
Duration Recall	0.9830	**0.9899**	0.9891	0.9796	0.9743
Duration F1	**0.9869**	0.5519	0.9834	0.9777	0.9715

Table A.1: Objective evaluation results on reconstruction criteria. PianoTree is our proposed method. Other columns correspond to the baseline models described in Section A.4.2.

A.4.4 Objective Evaluation of Reconstruction

The objective evaluation is performed by comparing different models in terms of their reconstruction accuracy of pitch onsets and note durations [91, 201], which are commonly used measurements in music information retrieval tasks. For note duration accuracy, we only consider the notes whose onset and pitch reconstruction is correct. Table A.1 summarizes the results where we see that the PianoTree VAE (the 1st column) is better than others in terms of F1 score for both criteria.

A.4.5 Latent Space Visualization

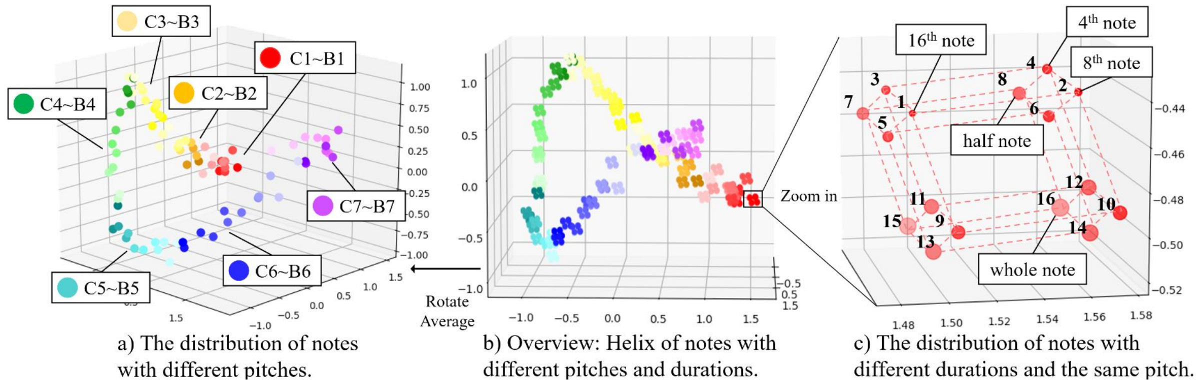
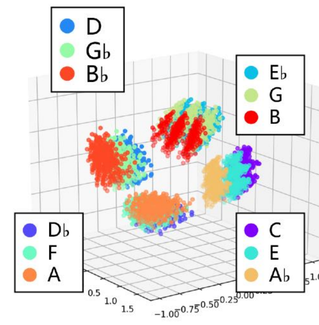


Figure A.4: A visualization of `note` embeddings after dimensionality reduction using PCA.

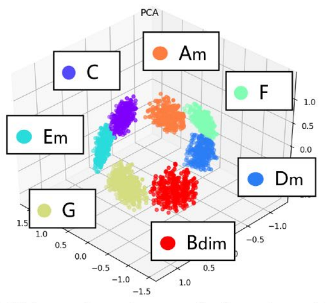
Figure A.4 shows the *latent note space* by plotting different `note` embeddings after dimensionality reduction by PCA (with the three largest principal components being reserved). Each colored dot is a `note` embedding, and a total of 1344 samples are displayed; note pitch ranges from C-1 to C-8, and note duration from a sixteenth note to a whole note.

We see that the `note` embeddings have the desired geometric properties. Figure A.4 (a) & (b) show that at a macro level, notes with different pitches are well sorted and form a “helix” in the 3-D space. Figure A.4 (c) further shows that at a micro level, 16 different note durations (with the same pitch) form a “fractal parallelogram” due to the binary encoding of duration attributes. One of the advantages of the encoding method is the translation invariance property. For example, the duration difference between the upper left cluster and the lower left cluster is 8 semiquavers, and so is the difference between the upper right and lower right clusters. The same property also applies to the four smaller-scale parallelograms.

Figure A.5 is a visualization of the latent chord space by plotting different `simu_note` embeddings under PCA dimensionality reduction. Each colored cluster corresponds to a chord label realized in 343 different ways (we consider all possible pitch combinations within three octaves, with a minimum of 3 notes and a maximum of 9 notes). The duration for all chords is one beat.



a) The relative relationship of different major chords.



b) Ring structure of chords with different degree scales in C major.

Figure A.5: A visualization of `simu_note` embeddings after dimensionality reduction using PCA.

The geometric relationships among different chords are consistent and human-interpretable. Specifically, Figure A.5 (a) shows the distribution of 12 different major chords, which are clustered in four different groups. By unfolding the circle in a counterclockwise direction, we can observe the existence of *the circle of the fifth*. Figure A.5 (b) is the visualization of seven C major triad chords: forming a ring in the order of 1-3-5-7-2-4-6 degrees in the counterclockwise direction.

A.4.6 Subjective Evaluation of Latent Space Interpolation

Latent space traversal [203, 271, 272] is a popular technique to demonstrate model generalization and the smoothness of the learned latent manifold. When interpolating from one music piece to another in the latent space, new pieces can be generated by mapping the representations back to the signals. If a VAE is well-trained, the generated piece will sound natural and form a smooth transition.

To this end, we invite people to subjectively rate the models through a double-blind online survey. During the survey, the subjects first listen to a pair of music, and then listen to 5 versions of interpolation, each generated by a model listed in Table A.1. Each version is a randomly selected pair of music segments, and the interpolation is achieved using SLERP[85]. Since the experiment requires careful listening and a long survey could decrease the quality of answers, each subject is asked to rate only 3 pairs of music, i.e., $3 \times 5 = 15$ interpolations in a random order.

After listening to the 5 interpolations of each pair, subjects are asked to select two best versions: one in terms of the *overall musicality*, and the other in terms of the *smoothness of transition*.

A total of n = 33 subjects (12 females and 21 males) with different music backgrounds have completed the survey. The aggregated result (as in Figure A.6) shows that the interpolations generated by our model are better than the ones generated by baselines, in terms of both overall musicality and smoothness of transition. Here, different colors represent different models (with the blue bars being our model and other colors being the baselines), and the height of the bars represents the percentage of votes (on the best candidate).

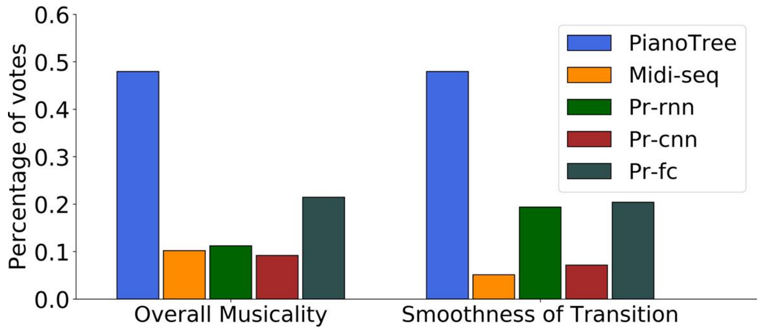


Figure A.6: Subjective evaluation results of latent space interpolation.

A.4.7 Downstream Music Generation

In this section, we further explore whether the polyphonic representation helps with *long-term music generation* when coupled with standard downstream sequence prediction models. (Similar tasks have been applied to *monophonic music* in [35] and [193].)

The generation task is designed in the following way: given 4 measures of piano composition, we predict the next 4 measures using a Transformer decoder (as in [241]). We compare three different music representations: MIDI-like event sequence (Section A.2.1), pretrained (decoder) `simu_note` embeddings, and latent vector z for every 2-measure music segment (without overlap). Here z is the mean of the approximated posterior from the encoder. For all three representations, we use the same Transformer decoder architecture (outputs of dimension = 128, number of layers

= 6, and number of heads = 8) with the same training procedure. Only the loss functions are correspondingly adjusted based on different representations: cross entropy loss is applied to midi-event tokens and MSE loss is applied to both `simu_note` and latent vector z. We use the same datasets mentioned in Section A.4.1 and cut the original piano pieces into 8-measure subsequent clips for generation purposes. We still keep 90% for training and 10% for testing.

We then invited people to subjectively rate different music generations through a double-blind online survey (similar to the one in Section A.4.6). Subjects are asked to listen to and rate 6 music clips, each of which contains 3 versions of 8-measure generation using different note representations. Subjects are told that the first 4 measures are given and the rest are generated by the machine. For each music clip, subjects rate it based on *creativity*, *naturalness*, and *musicality*.

A total of n = 48 subjects (20 females and 28 males) with different music backgrounds have participated in the survey. Figure A.7 summarizes the survey results, where the heights of bars represent means of the ratings and the error bars represent the confidence intervals computed via within-subject ANOVA [208]. The result shows that `simu_note` and latent vector z perform significantly better than the midi-event tokens in terms of all three criteria ($p < 0.005$).

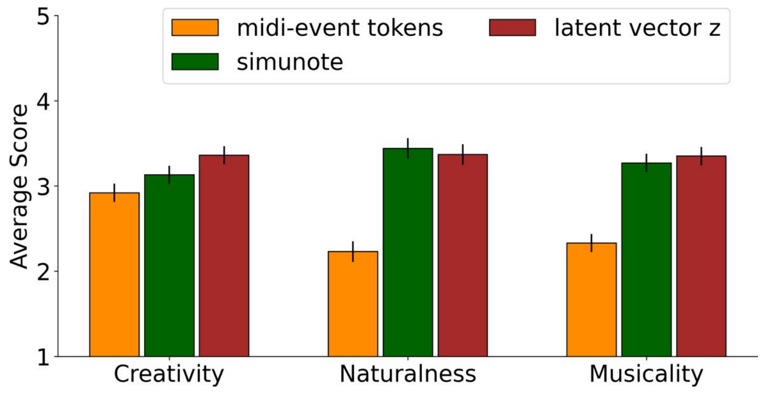
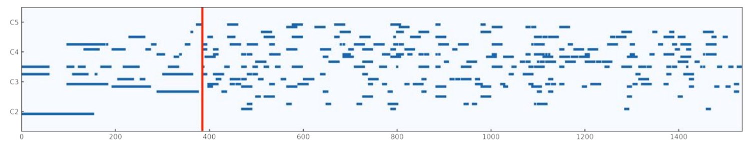
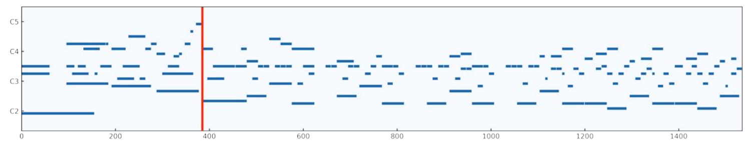


Figure A.7: Subjective evaluation results of downstream music generation.

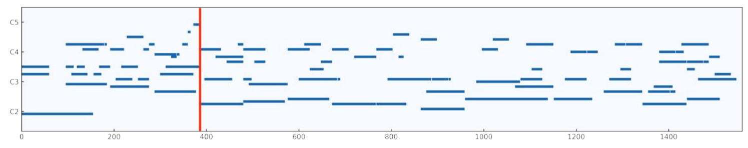
Besides the aforementioned generation task, we also iteratively feed the generated 4-measure music clips into the model to get longer music compositions. Figure A.8 shows a comparison of 16-measure generation results using all three representations. The first 4 bars are selected from the test set, and the subsequent 12 bars are generated by the models. Generally speaking, using `simu_note` and latent vector z as data representations yields more coherent music compositions. Furthermore, we noticed that long generations using the `simu_note` representation tend to repeat previous steps in terms of both chords and rhythms, while those generations using the latent vector z usually contain more variations.



(a) A sample generated using midi-event tokens.



(b) A sample generated using `simu_note`.



(c) A sample generated using latent vector z.

Figure A.8: Long music generations given first 4 measures.

A.5 Summary and Future Work

In summary, we proposed PianoTree VAE, a novel representation-learning model tailored for polyphonic music. The key design of the model is to incorporate both the music data structure and model architecture with the sparsity and hierarchical priors. Experiments show that with such inductive biases, PianoTree VAE achieves better reconstruction, interpolation, downstream generation, and strong model interpretability. In the future, we plan to extend PianoTree VAE for more general musical structures, such as motif development and multi-part polyphony.

B | Automatic Melody Reduction

This chapter is based on the published work *Automatic Melody Reduction via Shortest Path Finding* [246], conducted in collaboration with Yuxuan Wu, Roger B. Dannenberg, and Gus Xia. The algorithm proposed in this chapter is used for extracting the reduced lead sheet, the second-level hierarchical language of pop songs, as defined in Chapter 6.

Melody reduction, as an abstract representation of musical compositions, serves not only as a tool for music analysis but also as an intermediate representation for structured music generation. Prior computational theories, such as the Generative Theory of Tonal Music, provide insightful interpretations of music, but they are not fully automatic and usually limited to the classical genre. In this chapter, we propose a novel and conceptually simple computational method for melody reduction using a graph-based representation inspired by principles from computational music theories, where the reduction process is formulated as finding the shortest path. We evaluate our algorithm on pop, folk, and classical genres, and experimental results show that the algorithm produces melody reductions that are more faithful to the original melody and more musically coherent than other common melody downsampling methods. As a downstream task, we use melody reductions to generate symbolic music variations. Experiments show that our method achieves higher quality than state-of-the-art style transfer methods.[1]

[1]Music samples of melody reduction and variation can be found at https://auto-melody-reduction.github.io/AMRA-demo/. We release the code at https://github.com/ZZWaang/melody-reduction-algo.

B.1 Introduction

Maintaining structural coherence in long-term music generation is a fundamental challenge. One approach to addressing this challenge is through hierarchical models, which rely on extracting high-level *abstractions* to enable cascaded generative processes [57, 59, 243, 254]. These abstractions provide a coarser-grained view of musical structure, capturing essential long-range dependencies. In existing approaches, abstractions are typically explicitly defined (e.g., chord progression or phrase labels) or learned through unsupervised methods (e.g., latent codes via an autoencoder). Yet, so far, they have not been able to capture a fundamental musical structure: the *melodic flow*—how a melody evolves and resolves within a phrase—which remains too nuanced to be explicitly labeled and too challenging for unsupervised learning to reliably identify.

From a musicology perspective, melodic flow can be represented through *melody reduction*, which preserves the structural essence of a melody [141, 209]. However, most existing approaches regard melody reduction as a by-product of analysis, typically represented by hierarchical structures such as trees for further interpretation [186, 216]. In this context, reduction is not a fixed transformation but rather a subjective and demonstrative projection of the analysis procedure. This inherent ambiguity makes melody reduction not only difficult to evaluate but also challenging to use as a practical representation [81, 183, 233]. In this chapter, we explore how melody reduction can be approximated using structural heuristics, aiming to make the concept more accessible and useful for music generation.

To this end, we propose a fully automatic algorithm for melody reduction. The algorithm uses the graph representation of a melody phrase and regards all possible reductions as graph paths. The intuition behind the algorithm is that if we define a cost function consistent with guiding principles underlying most reduction theories, an ideal melody reduction should be the path having the least cost. Specifically, we consider two principles. First, the subsequent notes in an ideal melody reduction usually reveal a simpler structure (e.g., a prolongation (unison), or a linear progression (step-wise motion) [209]. Second, an ideal melody reduction usually includes notes of higher significance in terms of pitch, rhythm, and harmony [4, 141]. We define edge costs based on these principles and use the shortest-path algorithm to find the melody reduction [273]. The resulting path is then post-processed into an actual melody. The algorithm is not restricted to a specific genre, and it only requires annotations of MIDI pitches, onsets, durations, and the chord chromagram as the input.

We evaluate the proposed algorithm in pop, folk, and classical music genres, showing that it yields reductions that are often perceived as more faithful to the original melody and musically coherent compared to other melody downsampling methods. We also introduce variation generation as a downstream application, in which we train a melody generation model conditioned on reductions. The reductions extracted with our algorithm are shown to yield higher-quality variations compared to baselines.

B.2 Related Work

In this section, we review three realms of related work: 1) cognitive theories about music reduction, 2) the algorithmic implementation of music theories, and 3) the importance of melody reduction in downstream applications.

In the history of cognitive music theory, a shared methodology of music analysis is to use a reduced melody to represent the abstract melodic flow [141, 179, 209]. Schenkerian analysis involves a recursive reduction process to turn a music composition into the fundamental structure [209]; and the Generative Theory of Tonal Music (GTTM) further formalizes the grammar in Schenkerian analysis [141]. These studies highlight that melody reduction is an effective representation in the cognition process, and reduction is highly related to the considerations of note connection, harmonic context, pitch importance, etc., which usually imply tension and relaxation in different music scopes.

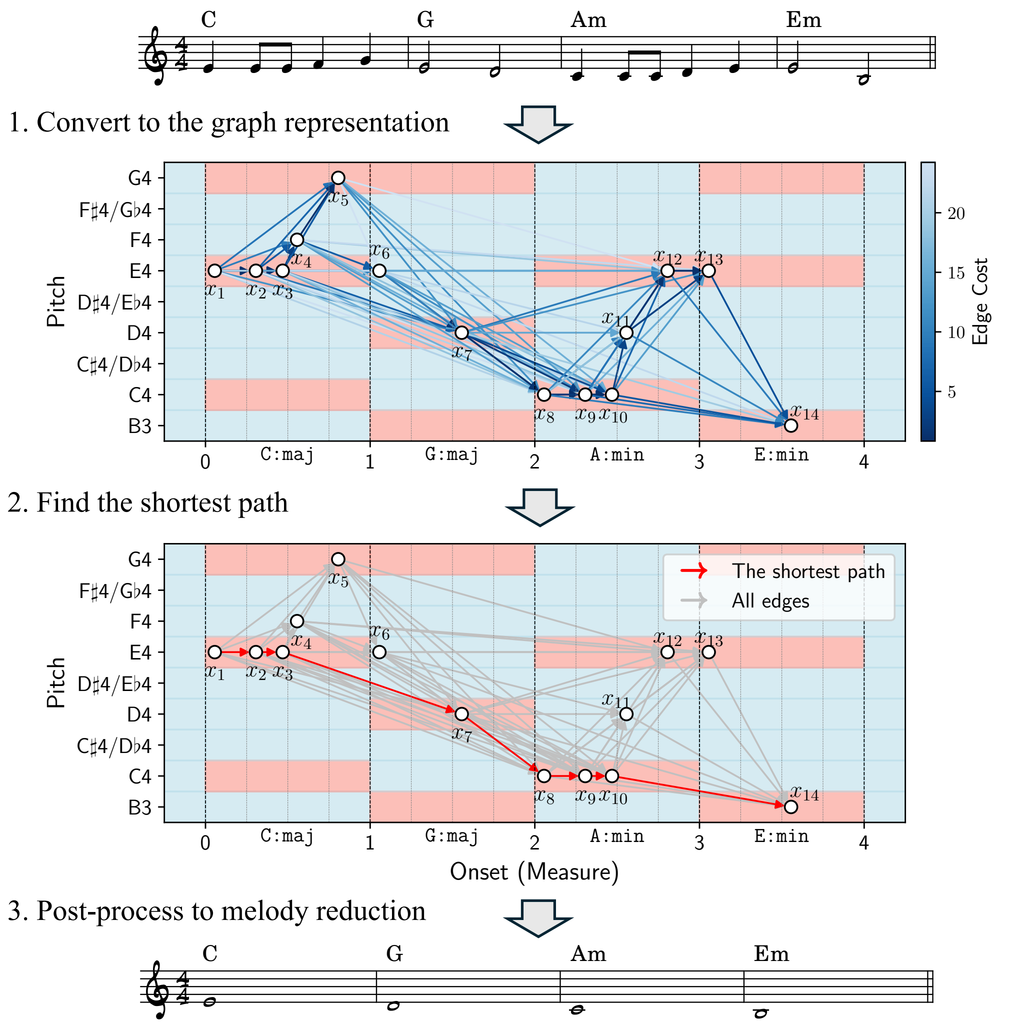

C
G
Am
Em
1. Convert to the graph representation
2. Find the shortest path
3. Post-process to melody reduction
Pitch
G4
F♯4/G♭4
F4
E4
D♯4/E♭4
D4
C♯4/D♭4
C4
B3
C:maj
G:maj
A:min
E:min
Onset (Measure)
Edge Cost
The shortest path
All edges
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14}

Figure B.1: The diagram of the proposed melody reduction algorithm.

There are several attempts to turn these theories into algorithms. Kirlin *et al.* propose a framework for automatic Schenkerian analysis [126], and Hamanaka *et al.* design an interactive software to implement GTTM using machine-learning techniques [87, 88, 233]. Other approaches reduce melodies recursively by assigning weights to notes [186, 216]. However, a quality gap remains between automatic analyses and human interpretations. Moreover, the algorithms usually require score-notation level data (e.g., MusicXML format), are genre-specific, and are not open-sourced. A recent computational music analysis points out that since our music preference is hard to express in formal grammar, such formal systems tend to have a broad search space of music analyses [70]. This motivates us to pursue an intuitive alternative: we directly approximate melody reduction based on cognitive preference without building a formal system.

Although melody reduction is mostly studied in an analytical scope, recent advances in deep learning also show that melody reduction representation is beneficial for structured long-term music generation. Previously, melody reduction was usually implicitly modeled by surrogate features such as down-sampled melody statistics [57], melody contour [36], or implicit latent representations [205]. The recent hierarchical music generation methodology shows that using an explicitly defined melody reduction, long-term music generation can be tackled more elegantly and effectively [243]. The algorithm we propose aims to establish a foundation for such future studies.

B.3 Methodology

In this section, we introduce the proposed melody reduction algorithm in detail. A diagram of our algorithm is shown in Figure B.1. Section B.3.1 introduces the data attributes and the graph representation of a melody. Section B.3.2 defines the edge types of the graph, and Section B.3.3 defines the edge cost. Finally, we discuss the melody reduction post-processing operations in Section B.3.4.

B.3.1 Graph Representation of a Melody

The input to the algorithm is a sequence of notes, denoted by $x_1, ..., x_N$, and an underlying chord progression, denoted by $c_1, ..., c_K$. A melody can be represented by a directed graph $\mathcal{G}(V, E)$, where the melody notes are regarded as graph nodes $V := \{x_i\}_{i=1}^{N}$, and temporal relations of notes can be represented by edges $E := \{x_i \to x_{i+1}\}_{i=1}^{N-1}$.

We consider the onset, pitch, and duration attributes of a note x_i. These are denoted by $\text{Onset}(x_i), \text{Pitch}(x_i)$ and $\text{Dur}(x_i)$, respectively. The onset and duration should be quantized by beat locations, and the pitch is represented by MIDI note numbers from 0 to 127. Additionally, a chord is represented by a 12-d binary chroma vector, i.e., $c_i \in \{0,1\}^{12}$, and we define $\text{Chord}(x_i) \in \{c_1, ..., c_K\}$ to indicate chord membership of the note x_i. In our algorithm, we heuristically detect anticipation-like cases (a type of non-chord tone) and regard these notes as belonging to the next chord.

B.3.2 Edge Definition

In the proposed algorithm, we regard both the original melody and reduced melodies as paths from x_1 to x_N. The original melody uses edges in E, whereas a reduction uses shortcut edges. To this end, we define an *augmented* edge set $E^* := \{x_i \to x_j | i < j\}$, representing all causal edges. If an edge $x_i \to x_j$ is selected in the reduction process, it means the melodic movement from $x_i \to x_j$ is more significant than all other movements $x_{i'} \to x_{j'}$ inside the time range, i.e., $i \leq i' < j' \leq j$ and $(i,j) \neq (i',j')$.

We categorize an edge $x_i \to x_j \in E^*$ into six categories. The first three categories are the most fundamental, which correspond to three main ways of melody reduction in Schenkerian analysis: prolongation, linear progression, and arpeggiation [31]. Note that the edges are strictly defined below, and we only borrow the terms for implication:

- **Prolongational Edge (PE)**: x_j prolongs x_i with the same pitch. For example, a PE can potentially remove a neighbor tone. Mathematically, a PE satisfies $\text{Pitch}(x_i) = \text{Pitch}(x_j)$ and $\text{Onset}(x_j) - \text{Onset}(x_i) < D$.

- **Linear Edge (LE)**: the interval between x_i and x_j is a second. For example, an LE can potentially mark a significant melodic movement. Mathematically, an LE satisfies $|\text{Pitch}(x_i) - \text{Pitch}(x_j)| \in \{1, 2\}$ and $\text{Onset}(x_j) - \text{Onset}(x_i) < D$.

- **Arpeggiation Edge (AE)**: the interval between x_i and x_j is larger than a (compound) second and x_i and x_j are within the same chord. For example, an AE can potentially mark an elaboration of harmony. Mathematically, an AE satisfies $|\text{PitchClass}(x_i) - \text{PitchClass}(x_j)| \in \{3, 4, 5, 6, 7, 8, 9\}$ and $\text{Chord}(x_i) = \text{Chord}(x_j)$.

In some melody compositions, pitches that span an octave are also regarded as a smooth connection. In Schenkerian analysis, this is explained by *imaginary continuo*—although the two tones span an octave in the current realization, they are close in other imaginary realizations. We define two types of imaginary edges accordingly:

- **Imaginary Prolongational Edge (IPE)**: x_j prolongs x_i with the same pitch class. Mathematically, an IPE (is not a PE) and satisfies $\text{PitchClass}(x_i) = \text{PitchClass}(x_j)$ and $\text{Onset}(x_j) - \text{Onset}(x_i) < D$.
- **Imaginary Linear Edge (ILE)**: the interval between x_i and x_j (or its inversion) is a compound second. Mathematically, an ILE (is not an LE) and satisfies $|\text{PitchClass}(x_i) - \text{PitchClass}(x_j)| \in \{1, 2, 10, 11\}$ and $\text{Onset}(x_j) - \text{Onset}(x_i) < D$.

Finally, all the rest of the edges in E^* belong to the final category. This is to ensure the graph is connected, so that there must exist at least one path from x_1 to x_N:

- **Unclassified Edges (UE)**: the rest of the edges.

In the above definition, $\text{Pitch}(\cdot)$, $\text{Onset}(\cdot)$, and $\text{Chord}(\cdot)$ are previously defined in Section B.3.1. PitchClass(x) := $\text{Pitch}(x) \mod 12$ and $\text{Chord}(x_i) = \text{Chord}(x_j)$ if and only if x_i and x_j are within the interval of a single chord. In our experiment, the temporal threshold D is set to 2 measures.

B.3.3 Edge Cost Definition

We define the edge cost function of $x_i \to x_j$ so that a more significant edge will have a smaller cost. The edge cost function considers three aspects: 1) the function of different edge types, 2) the temporal distance of an edge, and 3) the note importance.[2]

[2]Currently, the costs are empirically specified based on domain knowledge and preliminary analysis. Estimating them from data is left for future work.

First, we define *tonal cost*, denoted by $c_{\text{tonal}}(x_i \to x_j)$, prioritizing prolongational and linear edges in the melody reduction. Formally,

$$c_{\text{tonal}}(x_i \to x_j) := \begin{cases} 0.1, & \text{if } x_i \to x_j \text{ is a PE,} \\ 0.3, & \text{if } x_i \to x_j \text{ is an LE,} \\ 1.5, & \text{if } x_i \to x_j \text{ is an AE,} \\ 1.0, & \text{if } x_i \to x_j \text{ is an IPE,} \\ 1.3, & \text{if } x_i \to x_j \text{ is an ILE,} \\ 3.0, & \text{if } x_i \to x_j \text{ is a UE.} \end{cases} \tag{B.1}$$

Then, we define the *temporal cost*, denoted by $c_{\text{temp}}(x_i \to x_j)$, to measure the distance from index i to index j. We set the hyper-parameter $\eta = 1.6$ to achieve an ideal degree of reduction. A larger η results in too little reduction and a smaller η makes the reduction too coarse:

$$c_{\text{temp}}(x_i \to x_j) := (j - i)^{\eta}. \tag{B.2}$$

Besides the two cost functions on edges, we introduce a *note importance factor*, denoted by $\alpha(x_i)$, to ensure structurally important notes are more likely to be selected. Particularly, $\alpha(x_i)$ is a product of four terms:

$$\alpha(x) := \alpha_{\text{p}}(x)\alpha_{\text{o}}(x)\alpha_{\text{d}}(x)\alpha_{\text{h}}(x), \tag{B.3}$$

where $\alpha_{\text{p}}(x_i)$ denotes *pitch importance*, $\alpha_{\text{o}}(x_i)$ denotes *onset importance*, $\alpha_{\text{d}}(x_i)$ denotes *duration importance*, and $\alpha_{\text{h}}(x_i)$ denotes *harmony importance*.

1. **Pitch Importance.** Higher and lower pitches are usually more significant in a melody and

should be given a smaller weight factor:

$$\alpha_{\text{p}}(x_i) := 0.1 \times \left(0.5 - \frac{|\text{Pitch}(x_i) - p_{\text{mid}}|}{p_{\text{max}} - p_{\text{mid}}}\right) + 1, \tag{B.4}$$

where p_{max} and p_{min} are maximum and minimum pitch values and $p_{\text{mid}} = (p_{\text{max}} + p_{\text{min}})/2$.

2. **Onset Importance.** The notes having higher metrical importance should be given a smaller weight factor:

$$\alpha_{\text{o}}(x_i) := \begin{cases} 0.85, & \text{Onset}(x_i) \in \text{DB}, \\ 0.95, & \text{Onset}(x_i) \in \text{B}, \\ 1.05, & \text{Onset}(x_i) \in \text{B/2}, \\ 1.15, & \text{Onset}(x_i) \in \text{B/4}. \end{cases} \tag{B.5}$$

Here, DB, B, B/2, and B/4 represent downbeat, beat, eighth-note, and sixteenth-note positions, respectively (if under the 4/4 time signature).

3. **Duration Importance.** Longer notes should be given a smaller weight factor:

$$\alpha_{\text{d}}(x_i) := \begin{cases} 0.85, & \text{Dur}(x_i) \geq \text{half note}, \\ 0.95, & \text{Dur}(x_i) \geq \text{quarter note}, \\ 1.05, & \text{Dur}(x_i) \geq 8^{\text{th}} \text{ note}, \\ 1.15, & \text{Dur}(x_i) \geq 16^{\text{th}} \text{ note}. \end{cases} \tag{B.6}$$

4. **Harmony Importance**. A chord tone should be given a smaller weight factor than non-chord tones.

$$\alpha_{\text{h}}(x_i) := \begin{cases} 0.85, & x_i \text{ is a chord tone}, \\ 1.15, & \text{otherwise}. \end{cases} \tag{B.7}$$

Here x_i is a chord tone strictly means $\text{PitchClass}(x_i)$ is in $\text{Chord}(x_i)$. So, an anticipation is regarded as a chord tone to the next chord (see Section B.3.1).

Finally, the total edge cost is defined as a summation of tonal and temporal cost, modulated by the note importance factor:

$$c(x_i \to x_j) = \alpha(x_j)[c_{\text{temp}}(x_i \to x_j) + c_{\text{tonal}}(x_i \to x_j)]. \tag{B.8}$$

Thus, the melody reduction can be achieved by running a shortest-path algorithm to find the shortest path from x_1 to x_N.

B.3.4 Post-Processing

After we find the shortest path, we use a rule-based post-processing method to arrange the selected notes in the path to melody reduction. The maximum resolution of the reduction is a quarter note, in the style of a fifth-species counterpoint [49].

Figure B.2 shows the detailed procedure. First, the nodes in the shortest path are allocated into *chord bins*, with each bin corresponding to a distinct chord. In each chord bin, the notes are given a fixed rhythm template (see the table at the bottom of Figure B.2), ensuring the notes within a bin collectively span the entire duration of their associated chord. In this process, notes linked by a prolongational edge are merged into a single note. If the number of nodes in a chord bin exceeds the length of the chord, a random selection of notes will be omitted. Finally, the prolongational edges between two chords are marked with suspension. Note that notes serving as anticipations are allocated to the bin of the subsequent chord.

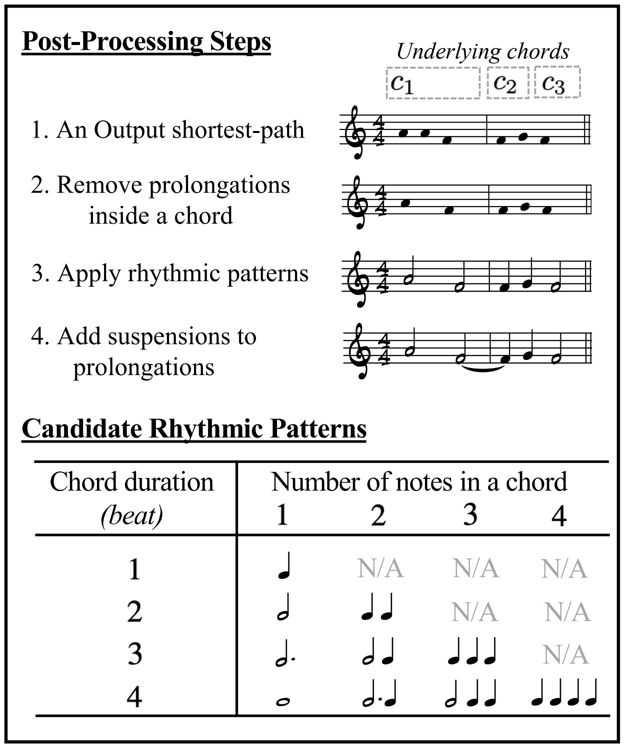

Post-Processing Steps
Underlying chords
c_1 c_2 c_3
1. An Output shortest-path
2. Remove prolongations inside a chord
3. Apply rhythmic patterns
4. Add suspensions to prolongations
Candidate Rhythmic Patterns
Chord duration (beat)
Number of notes in a chord
1 2 3 4
1 2 3 4
N/A

Figure B.2: An illustration of post-processing operations.

B.4 Experiments

In Section B.4.1, we evaluate the proposed algorithm through a subjective listening test. In Section B.4.2, we show and evaluate a melody reduction example as a case study. Finally, we evaluate the effectiveness of melody reduction in downstream music generation tasks in Section B.4.3.

B.4.1 Subjective Evaluation of Melody Reduction

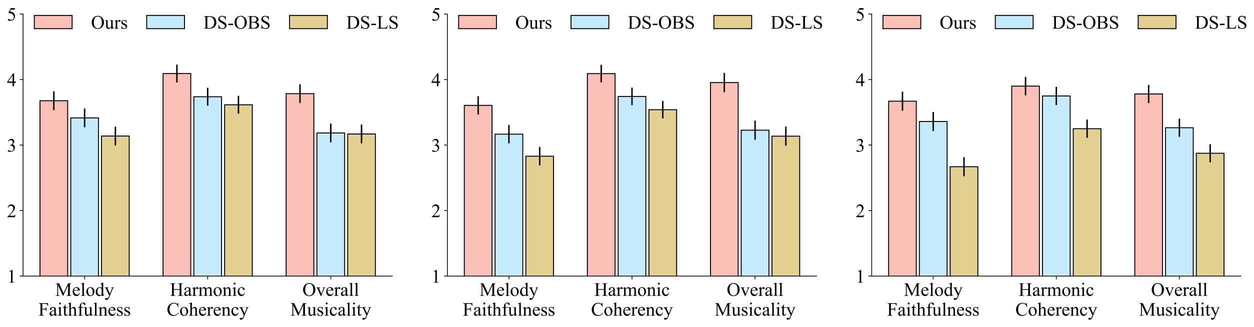
Unlike tasks with clear ground truths, melody reduction is inherently subjective and style-dependent. Existing theories, such as GTTM or Schenkerian analysis, provide interpretive hierarchies rather than prescriptive outcomes [31, 141]. Finding reduction typically involves pruning a tree at variable depths, often informed by human judgment. Moreover, such theories are primarily suited to classical music.

Given these challenges, we adopt a subjective listening test to better capture the perceptual and musical quality of melody reductions. We cover three music genres: pop, folk, and classical. We sample melodies from the POP909 dataset [249], the Nottingham dataset [72], and the GTTM database [86] for the pop, folk, and classical genres, respectively. We compare with two representative baselines commonly used for melody reduction as feature extraction in music generation:

- **Downsampling on Observations** (*DS-OBS*): From a statistical perspective, the melody is downsampled to a sequence of half notes, each representing the most common pitch in the 2-beat music segment [57].
- **Downsampling in Latent Space** (*DS-LS*): EC^2-VAE [271] learns disentangled latent representations of the pitch contour and rhythmic pattern of 2-measure music segments as z_p and z_r, respectively, which enables downsampling in the latent space of rhythm patterns. Specifically, we encode the pitch contour z_p of data and decode it together with a downsampled rhythm z_r to get the melody reduction for every 2-measure segment.

For the subjective test, we randomly select four 8-measure melodies from each genre. Each participant listens to at least 3 groups of melody reductions for each genre. In each group, participants are presented with the original melody first, followed by the melody reductions generated by the proposed algorithm, downsampling, and latent representation recombination in a randomized order. Participants are asked to rate the quality of the melody reduction on a 5-point Likert scale, where 1 indicates the worst quality and 5 indicates the best, in terms of three criteria: (1) *Melody Faithfulness*: how well the melody reduction preserves the original music information. (2) *Harmonic Coherency*: how well the melody reduction fits the underlying chord progression. (3) *Overall Musicality*: the overall music quality of the melody reduction.

A total of 45 subjects (26 females & 19 males) participated in the survey, in which over 70% have a music education experience of at least 2 years. The results are reported in Figure B.3, where the heights of bars represent means of the ratings and the error bars represent the standard error computed by within-subject ANOVA [208]. The results indicate that the proposed algorithm is



(a) Pop Genre **(b)** Folk Genre **(c)** Classical Genre

Figure B.3: Subjective evaluation results of melody reduction quality across three genres.

preferred over the two baseline methods in all three genres ($p_{\text{pop}} < 0.075$, $p_{\text{folk}} < 0.05$, $p_{\text{classical}} < 0.05$).

B.4.2 A Case Study of Melody Reduction

We provide a case analysis of melody reduction comparing the proposed algorithm and baselines introduced in Section B.4.1, as shown in Figure B.4. The original melody is shown in the top row, followed by the three melody reductions generated by the proposed algorithm and baselines.

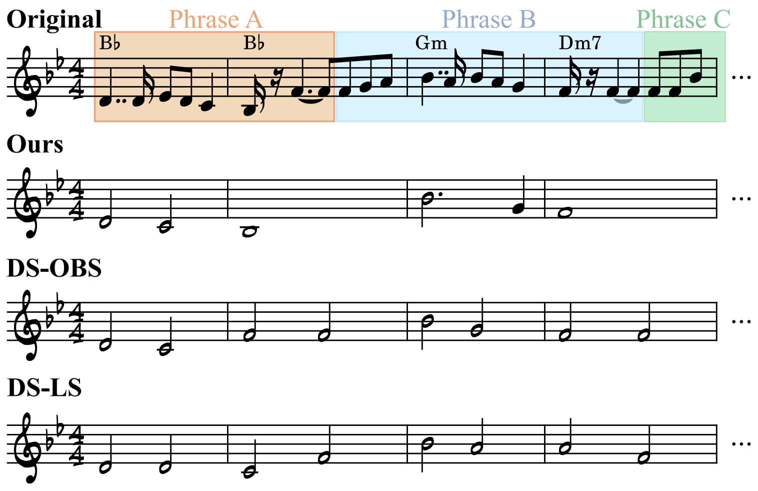


Figure B.4: Comparison of the original melody, melody reductions from the proposed method, and the baselines. We highlight the phrases in the top row.

Both the proposed method and DS-OBS can mostly capture the correct melody flow, such as important passing tones like C4 in the first measure. In subtle situations such as the second measure, where Phrase A ends its downward music flow with the downbeat chord tone B♭3 and lingers at F4 until the transition to Phrase B, DS-OBS fails to preserve B♭3, as it is overshadowed by the long duration of F4. In contrast, the proposed algorithm successfully captures B♭3 by paying attention to its harmonic and rhythmic importance, as well as the imaginary prolongational edge between B♭3 and B♭4 in the third measure. DS-LS only captures the pitch contour but introduces several unwanted non-chord tones. This example demonstrates the effectiveness and robustness of the proposed algorithm for melody reduction.

B.4.3 Downstream Task: Generating Melody Variations

We believe melody reduction can serve as a useful representation of structural information in downstream tasks. In this section, we demonstrate one such application in a melody variation generation task. The task uses the reduction of a melody as input, and outputs variations faithful to the original melody. While we do not claim a strong causal link between reduction quality and generation quality, our intuition is that an accurate reduction better reflects the underlying melodic and harmonic context, which in turn supports more coherent and musically grounded generation.

To this end, we train a diffusion model to generate melody variations from melody reductions provided by the proposed algorithm. We use a similar model design and training settings as the leadsheet generation model in [243] and train the model on the POP909 dataset. Similarly, we train the model using melody reductions by DS-OBS. We also generate melody variations by sampling in the latent space of z_p and z_r of EC^2-VAE for comparison. For all methods, we randomly sample four outputs per input and select the most representative one for use in listening tests.

Figure B.5 shows a group of melody variation examples. It can be seen that the variation model trained with the proposed melody reductions not only maintains the original melody flow but also

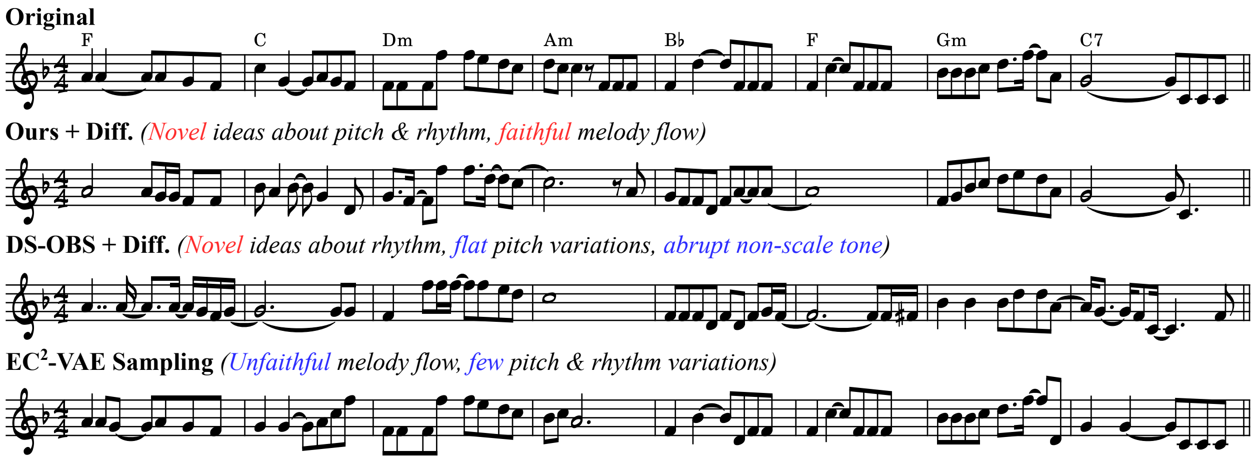

Original
F
C
Dm
Am
B♭
F
Gm
C7
Ours + Diff. (Novel ideas about pitch & rhythm, faithful melody flow)
DS-OBS + Diff. (Novel ideas about rhythm, flat pitch variations, abrupt non-scale tone)
EC²-VAE Sampling (Unfaithful melody flow, few pitch & rhythm variations)

Figure B.5: Comparison of variations of an example melody. Here **Diff.** denotes the conditional diffusion model trained to generate melody variations from melody reductions. Positive comments are highlighted in red, negatives in blue.

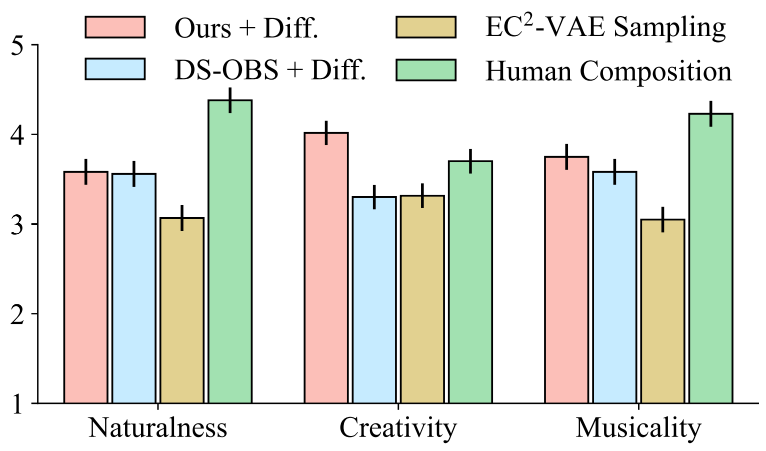


Figure B.6: Subjective results of melody variations.

introduces novel ideas in pitch and rhythm. The model trained with DS-OBS also preserves the pitch contour, but tends to have flat pitch variations. The variation generated by sampling from the latent space of EC^2-VAE changes the original melody flow in an unwanted way, and does not introduce rich variations.

We evaluate the melody variations on the test set of POP909 using a subjective listening test with the same participants as in Section B.4.1. Each participant listens to at least three groups of melody variations, where participants are first presented with the original melody, followed by the three variations in random order. Participants are asked to rate the quality of melody variations and the original melody by human composers in three criteria: *Naturalness*, *Creativity*, and *Musicality* [46]. The results are reported in Figure B.6, with the same computation as in Section B.4.1, including mean ratings and statistical significance tests. Our method is consistently preferred over two baselines in terms of creativity and overall musicality ($p < 0.05$), and remains competitive in naturalness.

B.5 Summary

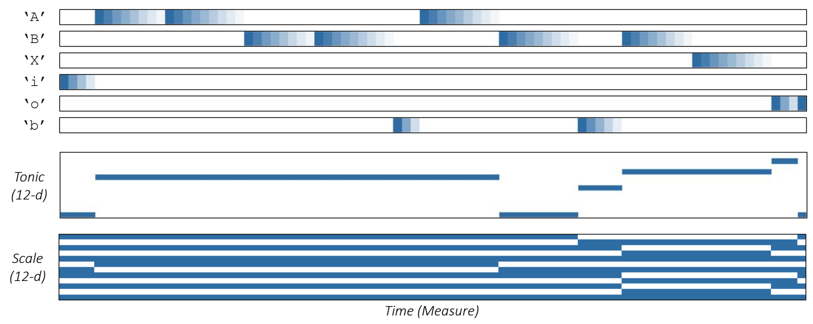
To sum up, this chapter proposed a novel and useful algorithm for melody reduction, filling the gap between the need to capture melody flow for long-term and hierarchical music generation and the lack of all-genre off-the-shelf tools for melody reduction. The proposed algorithm finds the optimal melody reduction by finding the shortest path in a graph representation of the melody, considering the tonal, temporal, and note importance factors. Subjective experiments demonstrated that our method outperforms baselines in a variety of musical styles. We also demonstrated the effectiveness of the melody reduction algorithm in melody variation generation through subjective evaluation. In the future, we plan to tackle reduction that captures latent polyphony and hierarchical structure, and explore the application of the proposed algorithm in a broader range of music generation tasks. While the current algorithm contains ad-hoc parameters, future work could also explore learning these directly from data.

C | Supplementary Materials for Chapter 6

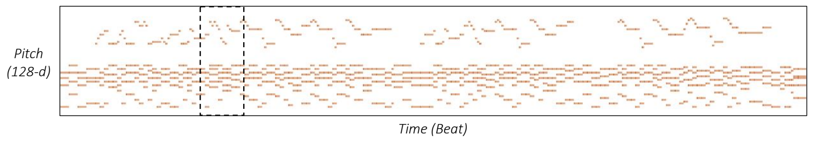
This chapter provides more technical details, generation samples, and discussions about Chapter 6. They are organized into three parts. Sections C.1–C.2 introduce more detailed information about data data representation and processing, model architecture, and training; Section C.3 provides some more generation samples; and Sections C.4–C.7 include more discussions on the external control efficacy, model's memorization v.s. generation ability, model efficiency, and limitations.

C.1 Data Representation and Processing Details

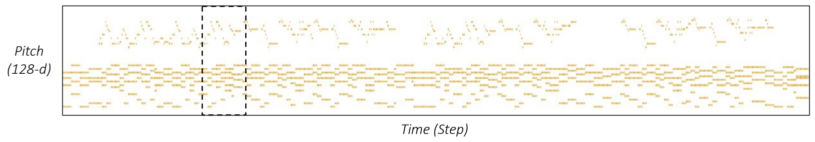
In this section, we introduce the methods to extract the four levels of music languages defined in Section 6.2.1. We begin by quantizing MIDI files from the POP909 dataset using the provided beat annotations. For the Form language, we simplify the phrase labeling from [55] to include only six phrase types (see Table C.1 for reference), and extract keys by pitch profile matching, similar to the method described in [134]. In the Reduced Lead Sheet language, melody reduction is computed through a shortest path reduction algorithm discussed in Chapter B, and chords are downsampled by merging consecutive chords that share the same triad structure (root, third, and fifth) within one measure. The Lead Sheet language uses the vocal melody (i.e., "MELODY" track)



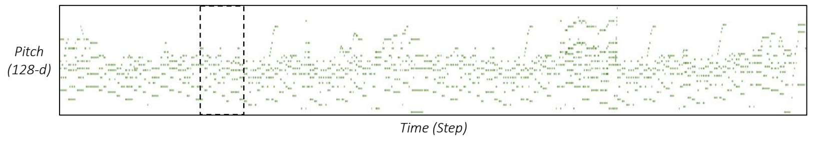
(a) The first-level language Form. The six channels of phrases are shown at the top and brightness indicates phrase countdown (see Equation 6.1). The two channels of keys are shown at the bottom and brightness indicates one-hot tonic and multi-hot scale.



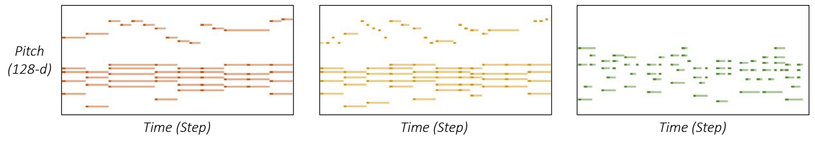
(b) The second-level language Reduced Lead Sheet. Brightness indicates onset and sustain channels.



(c) The third-level language Lead Sheet. Brightness indicates onset and sustain channels.



(d) The fourth-level language Accompaniment. Brightness indicates onset and sustain channels.



(e) Zoom-in views of the segments in Figure C.1(b) (left), Figure C.1(c) (middle), and Figure C.1(d) (right) marked with rectangles.

Figure C.1: An example data representation of our proposed hierarchical music language.

and the provided chord annotations. The Accompaniment language combines the secondary melody (i.e., “BRIDGE” track) with the accompaniment (i.e., “PIANO” track).

As discussed in Section 6.2.2, the processed data is converted to image-like data representation. A visualization of the data representation is provided in Figure C.1.

Phrase Type	**Channel ID**	**Meaning**
`"A"`	0	Verse section phrases
`"B"`	1	Chorus section phrases
`"X"`	2	Other phrases with lead melody
`"i"`	3	Intro section phrases
`"o"`	4	Outro section phrases
`"b"`	5	Bridge section phrases

Table C.1: Definition of phrase type

C.2 Model Architecture and Training Details

In this section, we elaborate on the model architecture and training. We first discuss the three conditioning methods introduced in Section 6.2.3 in more detail. The configurations of the four generation stages are summarized in Table C.2.

Detail on background condition. At each level $k > 1$, the background condition $\boldsymbol{X}^{<k}_{t:t+b_k}$ is represented by an image having the same width and height as the diffusion output. Thus, the background condition can be concatenated with the input along channel axis at each step of the diffusion process. The background condition will be set to all -1.0 under the probability $p_{\text{uncond}} = 0.2$, following classifier-free guidance [14].

Detail on autoregressive condition. At each level $k > 1$, the generation of $\boldsymbol{X}^{k}_{t:t+b_k}$ is dependent on past generation $\boldsymbol{X}^{\leq k}_{<t}$. Here we select top-$S_k$ past segments of length b'_k based on their *phrase type similarity* to the current segment. These music segments are embedded using a 3-layer 2d-convolutional network and fed to the backbone diffusion models by cross attention mechanism. In our implementation, we set $S_2 = 3$, $b'_2 = 32$, $S_3 = S_4 = 2$, and $b'_3 = b'_4 = 64$. The

	Form	**Red. Lead Sheet**	**Lead Sheet**	**Accompaniment**
Time scope	256 measures	128 beats	128 steps	128 steps
Output shape	$(6, 256, 12)$	$(2, 128, 128)$	$(2, 128, 128)$	$(2, 128, 128)$
Background cond.: shape	N/A	$(6, 128, 128)$	$(8, 128, 128)$	$(10, 128, 128)$
Background cond.: p_{uncond}	N/A	0.2	0.2	0.2
Autoreg. cond.: # of seg.	N/A	3	2	2
Autoreg. cond.: shape	N/A	$(8, 32, 128)$	$(10, 64, 128)$	$(12, 64, 128)$
Autoreg. cond.: p_{uncond}	N/A	0.1	0.1	0.1
Ext. cond.: # of latent codes	N/A	4	4	4
Ext. cond.: latent dimension	N/A	512	128	256
Ext. cond.: p_{uncond}	N/A	0.2	0.2	0.2

Table C.2: Configuration of the conditioning methods in four stages of the proposed model.

Hyperparameter	**Configuration**
Diffusion Steps (N)	1000
Noise Schedule	Linear from 1 to $1\text{e}-4$
UNet Channels	64
UNet Channel Multipliers	$1, 2, 4, 4$
Batch Size	16
Attention Levels	$3, 4$
Number of Heads	4
Learning Rate	$5\text{e}-5$

Table C.3: The hyperparameter configuration of diffusion model training. The listed attributes are the same across all four stages.

autoregressive condition will be set to all -1.0 under the probability $p_{\text{uncond}} = 0.1$.

Detail on external condition. The condition for Reduced Lead Sheet is four 8-measure latent codes of chord progression encoded from the chord encoder in [174]. The condition for Lead Sheet is four 2-measure latent codes of rhythmic pattern encoded from the EC^2-VAE encoder in [271]. The condition for Accompaniment is four 2-measure latent codes of accompaniment texture encoded from the texture encoder in [247]. These latent codes are fed to the backbone diffusion models by cross attention mechanism and set to all -1.0 under the probability $p_{\text{uncond}} = 0.2$.

The diffusion models for all four stages use the same noise schedule and training methods. Similar to [174], the backbone model is a 2D-UNet model, the encoder and decoder of which contain 4 layers of 2d-convolution with spatial attention at the third and fourth layers. We summarize these common details in Table C.3.

C.3 More Examples on Structural Generation

In this section, we break down each level of the hierarchical language and show more generation examples. For each level, we fix the upper level and demonstrate a variety of generation results under the upper level control. We also show generation samples that are controlled by external conditions.

Form generation. Below shows examples of Form generated by our model:

```
(i8)(A8B16A8B16)(b6)(B14)(o2)
(i12)(A4A4B12)(b4b4)(A4A4B12)(b4b4(B16)o4o1)
(i4)(A4A4B4X5)(b4)(A4b5B4X4X5)(o2)
(i4)(A8B9A8B9X18)(o2o1)
(i8)(A4A4B4B4)(b7)(A4A4B4B4B4B4)(o4)
(i8)(A4A4B4B4)(b7)(A4A4B4B4B4B4)(o4)
(i8)(A8B8X8X8)(b4b4)(A8B8X8X8X4)(o6o1)
(i4)(A4A4B9)(b3)(A4B10B9X1)(o2)
(i4)(A4A4B9)(b3)(A4B10B9X1)(o2)
(i12)(A16B16)(b4b4b4)(A16B16B16)(o10o1)
```

Here, we use parentheses to manually group music sections for better readability. The results show the model captures verse-chorus form of pop songs: the composition usually starts with intro and ends with outro; verses and choruses appear multiple times with bridge phrases in between. Phrases are usually 4 or 8 measures long, similar to real music samples.

Reduced Lead Sheet generation with external harmony control. Figure C.2a-f show examples of 8-measure generation of the Reduced Lead Sheet level. The results are all controlled

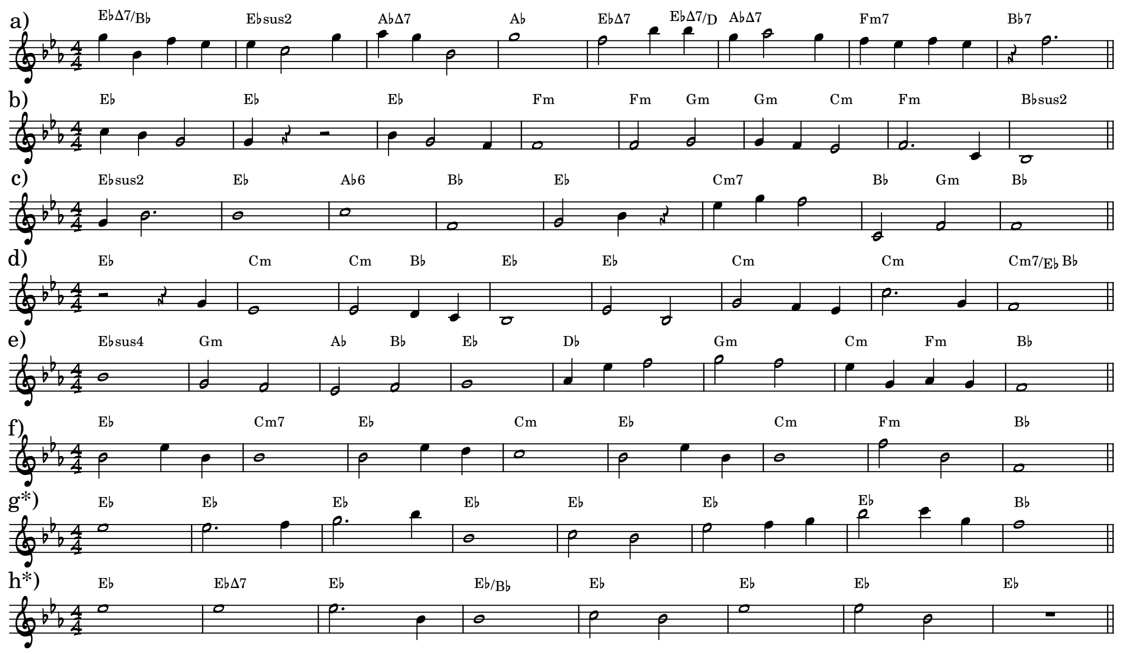


Figure C.2: Examples of generated Reduced Lead Sheet of "A8" phrase in E♭ major. The samples marked with * are controlled by the external condition, which represents "unchanging chord progression."

by the same Form: an 8-measure verse phrase in E♭ major key. The generated samples show many ways to develop the melody (different contour and melodic climax positions) and the harmony (different chord types and harmonic rhythm). Moreover, each of the samples has a consistent style and usually ends in a tonic or dominant chord, indicating the ending of a phrase. Moreover, we also apply the external control of "unchanging chord" to generate Figure C.2g-h. Such a control is achieved by encoding the latent chord representation of a sequence of all `Eb:maj` chords using the pre-trained VAE encoder [174]. The results have fewer changes in harmony, and the melody reduction alters accordingly.

Lead Sheet generation with external rhythm control. Figure C.3b-g show examples of 8-measure Lead Sheet generation controlled by the same Reduced Lead Sheet, shown in Figure C.3a. The generated samples follow the pitch contour in the melody reduction and differ in local pitch and rhythm patterns. At this level, we also use latent control of "sixteenth-note rhythm" to generate Figure C.3h-i. Such a control is achieved by encoding the latent rhythm representation

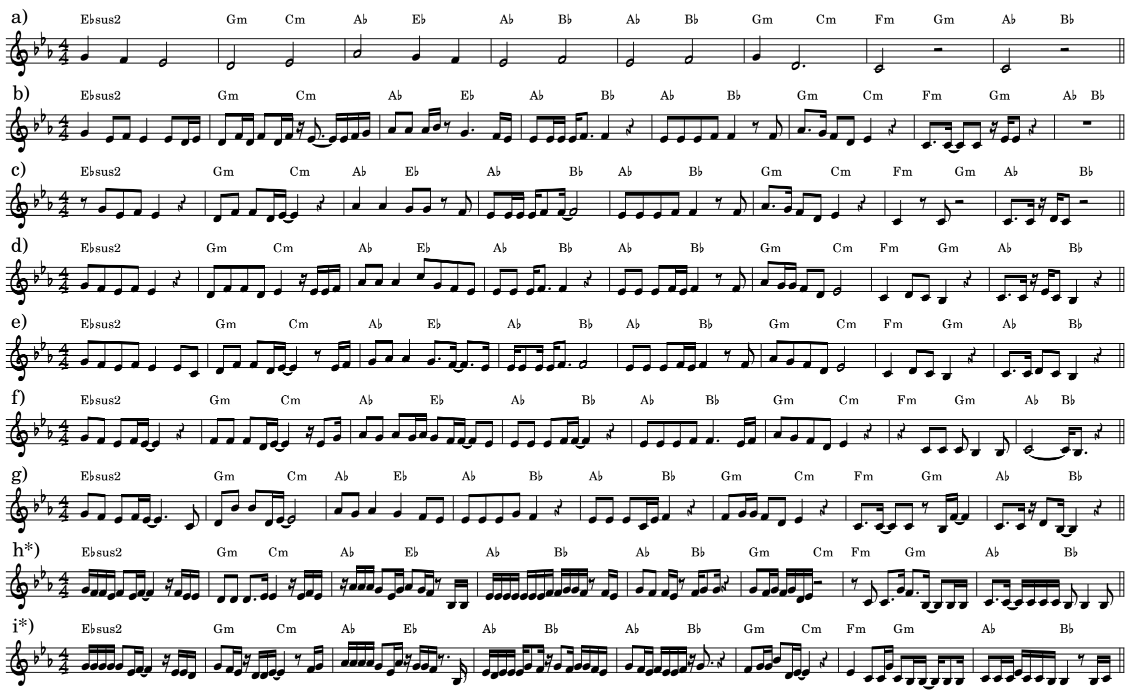

a) E♭sus2 Gm Cm A♭ E♭ A♭ B♭ A♭ B♭ Gm Cm Fm Gm A♭ B♭
b) E♭sus2 Gm Cm A♭ E♭ A♭ B♭ A♭ B♭ Gm Cm Fm Gm A♭ B♭
c) E♭sus2 Gm Cm A♭ E♭ A♭ B♭ A♭ B♭ Gm Cm Fm Gm A♭ B♭
d) E♭sus2 Gm Cm A♭ E♭ A♭ B♭ A♭ B♭ Gm Cm Fm Gm A♭ B♭
e) E♭sus2 Gm Cm A♭ E♭ A♭ B♭ A♭ B♭ Gm Cm Fm Gm A♭ B♭
f) E♭sus2 Gm Cm A♭ E♭ A♭ B♭ A♭ B♭ Gm Cm Fm Gm A♭ B♭
g) E♭sus2 Gm Cm A♭ E♭ A♭ B♭ A♭ B♭ Gm Cm Fm Gm A♭ B♭
h*) E♭sus2 Gm Cm A♭ E♭ A♭ B♭ A♭ B♭ Gm Cm Fm Gm A♭ B♭
i*) E♭sus2 Gm Cm A♭ E♭ A♭ B♭ A♭ B♭ Gm Cm Fm Gm A♭ B♭

Figure C.3: Examples of generated Lead Sheet of "A8" phrase in E♭ major given the upper-level Reduced Lead Sheet Figure C.3a. The samples marked with * are controlled by the external condition, which represents "sixteenth-note rhythm."

of the sample in Figure C.5(a) using the pre-trained VAE encoder [271]. The generation examples show melody realization with more frequent onsets accordingly.

Accompaniment generation with external texture control. Figure C.4b-d show examples of 8-measure Accompaniment generation controlled by the same Lead Sheet, shown in Figure C.4a. The generated samples mainly use arpeggios but are different in the exact patterns. Some of the generation has a "fill" in the fourth and eighth measures to indicate phrasing. At this level, we also use latent control of "Alberti bass" to generate Figure C.4e. Such a control is achieved by encoding the latent texture representation of the sample in Figure C.5(b) using the pre-trained VAE encoder [247]. The generation adopts the Alberti bass figure and makes reasonable variations.

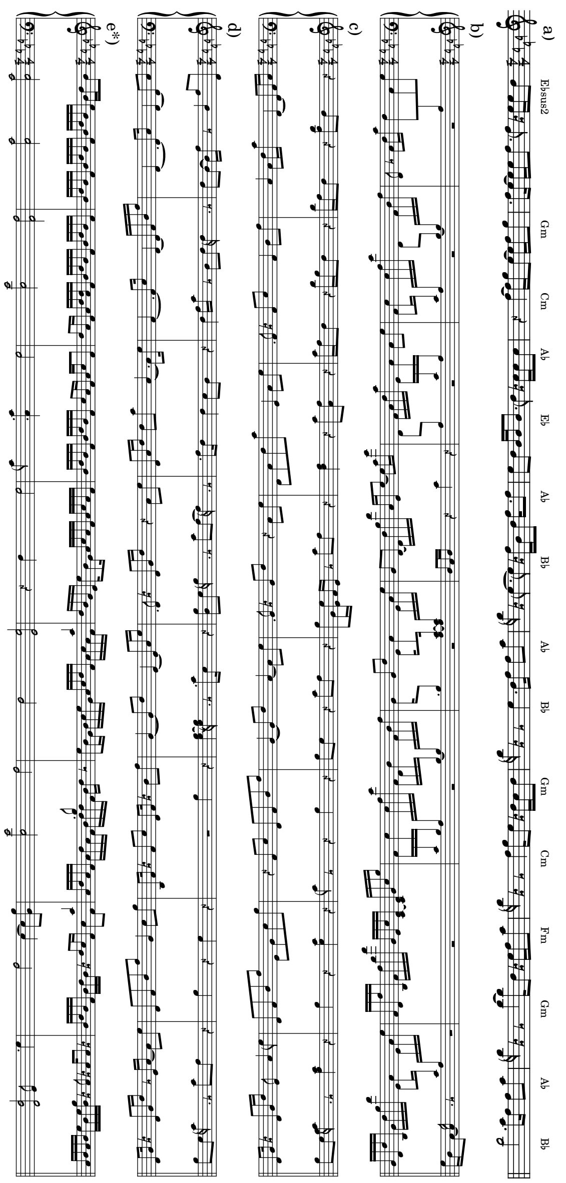
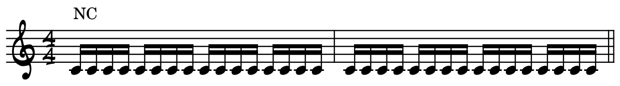
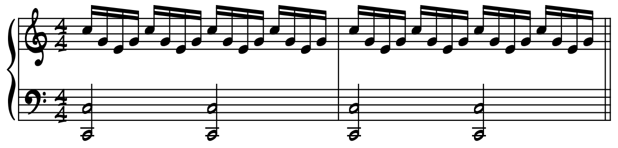


Figure C.4: Examples of generated Accompaniment of "A8" phrase in E♭ major given the upper-level Lead Sheet Figure C.4a. The sample marked with * is controlled by the external condition, which represents "Alberti bass."



(a) The external music sample used to control the generation of Figure C.3h-i.



(b) The external music sample used to control the generation of Figure C.4e.

Figure C.5: The music samples for external control.

C.4 Evaluation of External Control Efficacy

In this section, we evaluate the efficacy of external controls. These controls are achieved by feeding pre-trained representations as external condition to each layer of the cascaded diffusion model (introduced in Section 6.2.3). Specifically, we evaluate three scenarios: (1) chord control in Reduced Lead Sheet generation (Stage two), (2) rhythm control in Lead Sheet generation (Stage three), and (3) texture control in Accompaniment generation (Stage four). In this section, we let $\boldsymbol{z}^{\text{ext}}$ denote the external control in one of the three scenarios and let $\boldsymbol{x}^{\text{ext}}$ denote the actual observation from which $\boldsymbol{z}^{\text{ext}}$ is encoded. Let $\boldsymbol{x}^{\text{out}}$ denote the conditional generation results.

Efficacy of control can be evaluated by computing the similarity between the input control and the generation result. We propose a *rule-based metric* and a *latent metric*. The rule-based metric directly computes the distance between $\boldsymbol{x}^{\text{out}}$ and $\boldsymbol{x}^{\text{ext}}$ in terms of the corresponding features. In particular, for chord control, we compute the ℓ_2 distance between the given chord condition and the generated chord at each time step; and for rhythm or texture control, we compute the ℓ_2 distance of note onsets between the given control and the generated lead sheets or accompaniments. Such a distance-based metric has previously been used to evaluate control efficacy in [202] and [174]. In the latent metric, we encode the generation $\boldsymbol{x}^{\text{out}}$ back to the latent code $\boldsymbol{z}^{\text{out}}$ using the same pre-trained encoders and measure the cosine similarity between $\boldsymbol{z}^{\text{out}}$ and $\boldsymbol{z}^{\text{ext}}$.

There are two reference methods for comparison. First, we use the unconditional mode of our method to serve as a baseline where control is ineffective. Second, we generate samples by sampling from the Variational Autoencoders (VAEs) that the pre-trained encoders belong to. Specifically, we sample $\boldsymbol{z}^{\text{ext}}$ from the VAE posterior distribution and sample the rest of the latent codes from a Gaussian prior to decode results. By the well-disentangled property shown in the original paper [247, 271], this reference method indicates the maximum attainable level of controllability.

For each of the three scenarios, we randomly select 32 versions of external control from the test set and generate 128 music segments for each method. In Table C.4, we show the rule-based distance (denoted by dis^{rb}) and latent similarity (denoted by sim^{lt}) for the three generation stages. Experimental results show that the use of external condition significantly yields controllability for all three scenarios.

	Stage 1: Chord		**Stage 2: Rhythm**		**Stage 3: Texture**	
	$\text{dis}^{\text{rb}} \downarrow$	$\text{sim}^{\text{lt}} \uparrow$	$\text{dis}^{\text{rb}} \downarrow$	$\text{sim}^{\text{lt}} \uparrow$	$\text{dis}^{\text{rb}} \downarrow$	$\text{sim}^{\text{lt}} \uparrow$
Cas.Diff. (uncond)	2.09 ± 0.80	0.37 ± 0.09	2.27 ± 0.53	0.14 ± 0.23	3.94 ± 1.46	0.02 ± 0.11
VAE Sampling	0.19 ± 0.47	0.97 ± 0.07	0.14 ± 0.42	0.96 ± 0.04	0.33 ± 0.59	0.90 ± 0.06
Cas.Diff. (cond)	1.73 ± 1.02	0.48 ± 0.14	1.10 ± 0.74	0.75 ± 0.16	0.87 ± 0.80	0.89 ± 0.06

Table C.4: Objective evaluation of external control efficacy of chord, rhythm, and texture in the three diffusion stages. dis^{rb} denotes the rule-based distance-based metric and sim^{lt} denotes the latent similarity-based metric.

C.5 Does the Model Just Copy the Training Data?

In generative modeling, a critical consideration is whether the model overfits and the generation copies the training data. This section includes a quantitative evaluation focused on the similarity between generated segments and the entire training set. We primarily focus on *melody* similarity, a most recognizable aspect of music composition.

Our goal is to measure the *Degree of Copying (DoC)* with respect to a set of generated samples. Let $\boldsymbol{x}$ be a two-measure melody segment from a generated piece. We define *Similarity to the Training Set* of segment $\boldsymbol{x}$ as:

$$S(\boldsymbol{x}) := \max_{\boldsymbol{x}' \in \mathcal{T}} \operatorname{sim}(\boldsymbol{x}, \boldsymbol{x}'), \tag{C.1}$$

where $\mathcal{T}$ denotes the training set, $\boldsymbol{x}'$ is a two-measure segment from the training set, and $\operatorname{sim}(\boldsymbol{x}, \boldsymbol{x}')$ computes the similarity between $\boldsymbol{x}$ and $\boldsymbol{x}'$. Here $S(\boldsymbol{x}) \in [0, 1]$, and a larger $S(\boldsymbol{x})$ shows a higher degree of copying. The DoC can be represented by the histogram of $S(\boldsymbol{x})$. In our experiments, we report the mean and standard deviation of the histogram. We consider a rule-based similarity metric and a latent similarity metric as follows:

Rule-based similarity metric. We compute the note-wise similarity between two segments by matching the exact onsets and pitches. Let $n_{\boldsymbol{x} \cap \boldsymbol{x}'}$ denote the number of notes that appear in both $\boldsymbol{x}$ and $\boldsymbol{x}'$ with the same *pitch class* and *onset*, and let $n_{\boldsymbol{x}}$ and $n_{\boldsymbol{x}'}$ denote the number of notes in $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively. The rule-based similarity metric is defined as:

$$\operatorname{sim}^{\mathrm{rb}}(\boldsymbol{x}, \boldsymbol{x}') := \frac{2 n_{\boldsymbol{x} \cap \boldsymbol{x}'}}{n_{\boldsymbol{x}} + n_{\boldsymbol{x}'}}. \tag{C.2}$$

Latent similarity metric. We also measure the melodic similarity in the latent space because rule-based methods cannot detect *indirect copying* (e.g., same pitch contour or rhythm). We leverage the pre-trained EC^2-VAE [271], which learns a semantically meaningful and disentangled latent space of pitch contour and rhythmic pattern. We extract the latent code of pitch (denoted as $\boldsymbol{z}_{\mathrm{p}}^{\boldsymbol{x}}$) and rhythm (denoted as $\boldsymbol{z}_{\mathrm{r}}^{\boldsymbol{x}}$) of melody segments and compute the cosine similarity in terms of both pitch and rhythm:

$$\operatorname{sim}_{\mathrm{p}}^{\mathrm{lt}}(\boldsymbol{x}, \boldsymbol{x}') := \frac{\langle \boldsymbol{z}_{\mathrm{p}}^{\boldsymbol{x}}, \boldsymbol{z}_{\mathrm{p}}^{\boldsymbol{x}'} \rangle}{||\boldsymbol{z}_{\mathrm{p}}^{\boldsymbol{x}}|| \cdot ||\boldsymbol{z}_{\mathrm{p}}^{\boldsymbol{x}'}||}, \tag{C.3}$$

$$\operatorname{sim}_{\mathrm{r}}^{\mathrm{lt}}(\boldsymbol{x}, \boldsymbol{x}') := \frac{\langle \boldsymbol{z}_{\mathrm{r}}^{\boldsymbol{x}}, \boldsymbol{z}_{\mathrm{r}}^{\boldsymbol{x}'} \rangle}{||\boldsymbol{z}_{\mathrm{r}}^{\boldsymbol{x}}|| \cdot ||\boldsymbol{z}_{\mathrm{r}}^{\boldsymbol{x}'}||}. \tag{C.4}$$

For both of the metrics, the samples in the training set are transposed to 12 keys to account for relative pitch similarity. Segments that only contain rests are discarded beforehand.

Sample Source	**Sample Size**	**Similarity Metric**		
		$\mathrm{sim}^{\mathrm{rb}} \downarrow$	$\mathrm{sim}_{\mathrm{p}}^{\mathrm{lt}} \downarrow$	$\mathrm{sim}_{\mathrm{r}}^{\mathrm{lt}} \downarrow$
Test set *(no plag.)*	88 pieces	0.6567 ± 0.1141	0.8637 ± 0.0486	0.8320 ± 0.0680
Copy-bot 1 *(plag.)*	128 pieces	0.7108 ± 0.1159	0.8616 ± 0.0526	0.8276 ± 0.0699
Copy-bot 2 *(plag.)*	128 pieces	0.6888 ± 0.1628	0.9086 ± 0.0340	0.8555 ± 0.0411
Cas.Diff. (ours)	128 pieces	0.6530 ± 0.1321	0.8743 ± 0.0491	0.8180 ± 0.0710
Polyff. + ph.l.	128 pieces	0.6117 ± 0.1162	0.8639 ± 0.0487	0.8424 ± 0.0622
TFxl(REMI) + ph.l.	128 pieces	0.6088 ± 0.1053	0.8599 ± 0.0446	0.8154 ± 0.0642

Table C.5: Evaluation on whether the generative models copy the training data. The highlighted data in red indicates potential copying the training set.

In Table C.5, we show the mean and standard deviation of $S(\boldsymbol{x})$ on the data samples generated using our proposed methods and other baselines used for whole-song generation. We compute the statistics of the test set of POP909 as a reference for *no risk of copying*, since no song in the training set (or their cover-song versions) appears in the test set. We also design two copy-bots as references for *potential risk of copying*. The first copy-bot copies a different part of the training set at each measure, which emulates a *direct copying* behavior. The second copy-bot encodes the melodies from the training set and adds noise to the latent representation before reconstruction, which emulates an *indirect copying* behavior. *Experimental results show that our proposed method (as well as the baseline whole-song generation methods) have similar DoC compared to that of the test set. Also, the proposed metrics successfully detect both direct and indirect copying behaviors as the DoCs of copy-bots are noticeably higher. Thus, we conclude that our model has a very low risk of copying the training set.*

C.6 Discussion: Efficiency of the Cascaded Method

In this section, we conduct a theoretical comparison between the efficiency of our cascaded diffusion models and an alternative end-to-end approach, which generates a full-piece music with a single diffusion model. We evaluate the *time* and *model parameter* complexities of both methods. As summarized in Table C.6, we demonstrate that end-to-end approaches exhibit quadratic complexities in both time and model parameters relative to data length, whereas the cascaded approach achieves linear time complexity and logarithmic model parameter complexity.

We base our analysis on a diffusion architecture with a UNet backbone, identical to the proposed architecture. We assume the sequential data has length T (corresponding to image width in UNet), and the dimension of each time step is D (corresponding to image height in UNet). The UNet will first embed the input image to have shape (C, T, D) regardless of the number of input channels, where C is called the base channel size. Additionally, our computation assumes a typical UNet configuration as follows:

1. In the contracting path of the UNet, the number of channels at each layer doubles, and the width of the feature maps are halved due to max pooling.

2. The expanding path of the UNet mirrors the contracting path.

3. The number of the levels (or model depth) $M_{T'}$ scales logarithmically with the input sequence length T', i.e., $M_{T'} = \log_2 T' + c_0$, where c_0 is a constant.

Complexity of end-to-end approach. At each level $i = 0, \ldots, M_T = \log_2 T + c_0$, the width of the feature map is $\frac{T}{2^i}$, the number of input channels is $C \cdot 2^i$, and the number of output channels is $C \cdot 2^{i+1}$. The time complexity of the convolution per layer can be computed as:

$$\mathcal{O}\big((\frac{T}{2^i} \cdot D) \cdot (C \cdot 2^i) \cdot (C \cdot 2^{i+1})\big) = \mathcal{O}(T \cdot D \cdot C^2 \cdot 2^{i+1}) = \mathcal{O}(T \cdot 2^{i+1}). \tag{C.5}$$

In Equation C.5, we regard D and C as constants, therefore removing from the complexity term. Summing Equation C.5 over all levels results in the total time complexity:

$$\sum_{i=0}^{M_T} \mathcal{O}(T \cdot 2^{i+1}) = \mathcal{O}(T^2). \tag{C.6}$$

Similarly, at each level $i = 0, \dots, M_T$, the number of model parameters can be computed as:

$$\mathcal{O}\big((C \cdot 2^i) \cdot (C \cdot 2^{i+1})\big) = \mathcal{O}(C^2 \cdot 2^{2i+1}) = \mathcal{O}(2^{2i+1}), \tag{C.7}$$

resulting in the total model parameter complexity:

$$\sum_{i=0}^{M_T} \mathcal{O}(2^{2i+1}) = \mathcal{O}(T^2). \tag{C.8}$$

In conclusion, the time and model parameter complexity for end-to-end approach are both $\mathcal{O}(T^2)$.

Complexity of cascaded approach. The cascaded models proposed in Chapter 6 is tailored for music data. For the analysis in this section, we define a theoretical cascaded approach as follows. To generate a sequential data of length T, we define a K-level compositional hierarchy, and the resolution at each level is scaled by a factor η. The top-level ($k = 1$) has a resolution of L, and for each level $k = 2, \dots, K$, the resolution is $L \cdot \eta^{k-1}$, such that at the final level K, the resolution is $L \cdot \eta^{K-1} = T$. We train K diffusion models in total, all having the same generation scope L. Thus, the diffusion model at level $k = 1$ generates the whole sequence, and the diffusion models for level $k = 2, \dots, K$ generate only a slice of the sequence. The model parameter complexity of a single level can be computed by substituting M_T with M_L in Equation C.8:

$$\sum_{i=0}^{\log_2 L + c_0} \mathcal{O}(L \cdot 2^{i+1}) = \mathcal{O}(L^2). \tag{C.9}$$

Summing up the parameters in K separate models results in the total model parameter complexity:

$$\mathcal{O}(L^2 \cdot K) = \mathcal{O}(L^2 \log_\eta T). \tag{C.10}$$

Similarly, for time complexity, each model call is $\mathcal{O}(L^2)$ (see Equation C.6); and at all levels, the generation requires $(2\eta^{k-1} - 1)$ autoregressive iterations (see Algorithm 1). So, the total time complexity is:

$$\sum_{k=1}^{K} \mathcal{O}(L^2 \cdot \eta^{k-1}) = \mathcal{O}(L^2 T). \tag{C.11}$$

Therefore, the cascaded approach has a time complexity of $\mathcal{O}(L^2 T)$ and a model parameter complexity of $\mathcal{O}(L^2 \log_\eta T)$. The computation implies that the efficiency of the cascaded approach will be more significant when $T >> L$. Theoretically, if L is a constant (e.g., bounded by the computational resources available), the time and model parameter complexity will become $\mathcal{O}(T)$ and $\mathcal{O}(\log_\eta T)$, respectively.

	Cascaded Approach	**End-to-End Approach**
Time Complexity	$\mathcal{O}(L^2 T)$	$\mathcal{O}(T^2)$
Model Parameter Complexity	$\mathcal{O}(L^2 \log_\eta T)$	$\mathcal{O}(T^2)$

Table C.6: Theoretical comparison of time and model parameter complexity between the cascaded approach and the end-to-end approach. η denotes the resolution scaling factor. L denotes the time scope (i.e., receptive field) of cascaded models.

C.7 Limitations and Future Plan

Our current model supports a maximum generation scope of 256 measures, typically adequate for pop songs, but insufficient for other genres (e.g., classical music), which may require longer lengths. While our model can generate irregular phrase lengths and theoretically supports both 3/4 and 4/4 time signatures, it does not support meter change, and the capability of 3/4 song generation is limited, since the proportion of 3/4 songs in the dataset is pretty low. Additionally, we observe that the generated endings are sometimes not ideal, such as failing to resolve on the tonic harmony. We suspect the issue is related to imprecise quantization in the POP909 dataset's outro sections. Moreover, there is room for improvement in overall music quality, as the model occasionally produces flawed samples, such as blank measures. To address these issues, we plan to enhance model performance with a more refined architecture and more extensive data training.

D | Supplementary Materials for Chapter 8

This chapter provides more technical details, results, and discussions about Chapter 8 and is structured into 5 main parts. Section D.1 provides specifics about the datasets involved in the study. Section D.2 presents implementation and training details of V3 and baseline methods. Section D.3 provides additional experiment results and especially visualizations for better understanding. Section D.4 presents an ablation study on the V3 model. Finally, we provide an analysis on learning content and style in Section D.5.

D.1 Dataset Details

D.1.1 PhoneNums

The written phone numbers dataset is designed to represent a clear content and style separation to humans. We use the Kristen ITC font for the style because its digits look similar to handwritten digits and are easy to distinguish. We render the digits from 0 to 9 on a light background of RGB (10, 10, 10) using the foreground colors listed in Table D.1.

For more randomness, we first jitter the foreground and background colors by a noise from -2 to 2 along every channel, then add a small Gaussian noise. We then translate all digits vertically or horizontally by a random number of pixels between -2 and 2. Lastly, we add a random Gaussian

RGB Values #	Color
(8, 8, 8)	Black
(8, 8, 248)	Blue
(8, 128, 8)	Green
(248, 8, 8)	Red
(8, 128, 128)	Teal
(128, 8, 128)	Purple
(248, 163, 8)	Orange
(163, 47, 47)	Brown
(248, 188, 199)	Pink *
(243, 128, 115)	Salmon *
(248, 210, 8)	Gold *
(8, 248, 8)	Lime *
(8, 248, 248)	Cyan *
(248, 8, 248)	Magenta *
(128, 128, 128)	Gray *
(200, 133, 67)	Peru *

Table D.1: List of colors and their corresponding RGB values. Colors only used for out-of-distribution experiments in Section 8.3.3 are marked with *.

blur effect. Our dataset contains 100000 images in total, each of which has 10 digits. The dataset is split into the train set, validation set, and test set with a ratio of 8:1:1. Some samples of the dataset can be viewed in Figure D.1

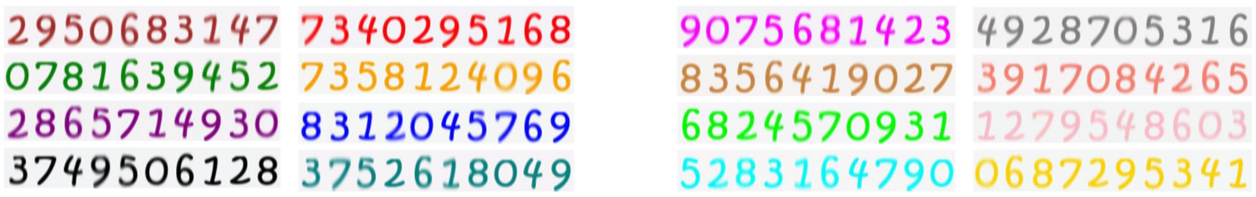


Figure D.1: Left: example training data in PhoneNums. Right: example data for out-of-distribution evaluation in PhoneNums.

D.1.2 InsNotes

In InsNotes, we collect monophonic music audio played by different instruments. This dataset is also designed based on the understanding that this domain exhibits content and style concepts that are clear to humans—music pitches and instrument timbres. The dataset consists of monophonic music audio rendered from 12 instruments playing 12 different pitches in an octave, which corresponds to MIDI numbers from 60 to 71. All the instruments selected have little exponential decay, so their timbres can be well represented with short audio samples. In the out-of-distribution generalization experiment in Section 8.3.3, we select four unseen instruments: two with slight exponential decays and two with strong attacks and distinct exponential decays, making the task particularly challenging. We list the instruments involved as well as the specific MIDI program selected in Table D.2.

Program #	**Instrument**
19	Pipe organ
21	Accordion
22	Harmonica
41	Viola
52	Choir aahs
56	Trumpet
59	Muted trumpet
64	Soprano sax
68	Oboe
71	Clarinet
72	Piccolo
75	Pan flute
0	Grand piano *
4	Tine electric piano *
73	Flute *
78	Irish flute *

Table D.2: List of instruments and their corresponding MIDI program numbers. Instruments only used for out-of-distribution generalization are marked with *.

For every instrument, we play every pitch for one second, one by one, with a random velocity between 80 and 120 until every pitch is played 10 times. We synthesize 100 such takes at 16kHz using a soundfont library for each instrument and further diversify every note by adding a random amplitude envelope to each note. The added amplitude envelope is either a linear curve or a sinusoidal curve, starting and ending at a random amplitude factor between 0.8 and 1.2. The audio files are then normalized and processed with short-time Fourier transform (STFT) with the FFT size of 1024 and hop size of 512 to obtain the magnitude spectrograms, which results in a 512×32 matrix for each note. To avoid possible overlap between adjacent notes, we add a 0.056-second pause in between, resulting in one transition frame in the spectrogram. Our dataset contains 1200 audio files in total, each of which has 120 notes. The dataset is split into the train set, validation set, and test set with a ratio of 8:1:1.

D.1.3 SVHN

The Street View House Numbers (SVHN) dataset is a real-world dataset that contains images of house numbers collected from Google Street View, and digit-level bounding boxes [182]. Examples of the dataset can be viewed in Figure D.2. The dataset originally consists of 73257 digits for training, 26032 digits for testing, and 531131 additional, somewhat less difficult samples, as the extra partition. We split the extra partition into additional training, validation, and testing sets with a ratio of 8:1:1. For the content-style disentanglement task, we select all images with at least two labeled digits, and resize the bounding boxes to 32×48 pixels. The dataset is preprocessed by normalizing the pixel values to the range of [0, 1]. Compared to PhoneNums, although both being image datasets with digits as content, SVHN is significantly more challenging for the following reasons: 1) The digits in SVHN can be very blurry compared to that in PhoneNums; 2) The digits in SVHN come with more flexible styles in a totally continuous space, involving different fonts, thicknesses, inclinations, colors, and so on; 3) In every SVHN image, the style variation among digits can be more significant than that in PhoneNums, as there can be environmental factors like shadows; 4) The bounding boxes are not always tight, clean and complete, and the digits are not always centered in the image; 6) The classes are very imbalanced. Almost all images come with a 0, 1, or 2, but very few of them have 8 or 9; 5) Most importantly, house numbers are generally very short strings. Among images with at least two digits, 57.6% of them have exactly two digits, which means for most styles, there is not full coverage of all digits, and during training, V3 only

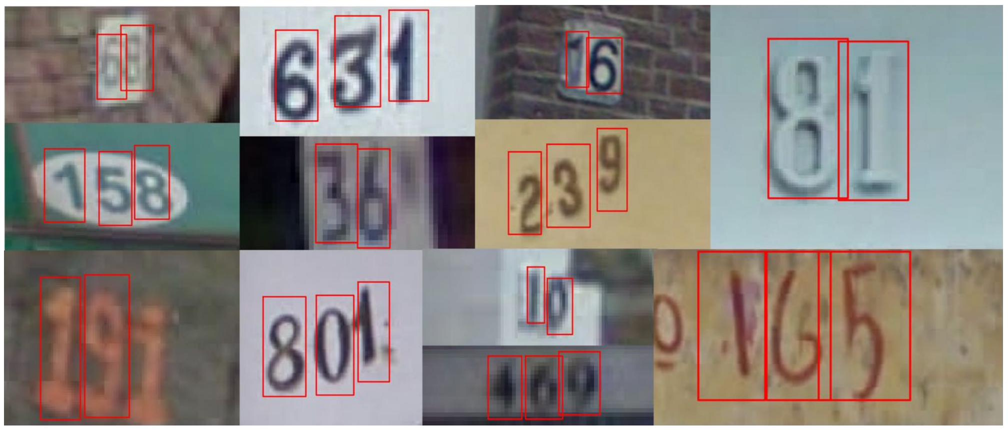


Figure D.2: Example data in the original SVHN dataset. The digits in the images are bounded by the red boxes.

has a highly incomplete view of the full content vocabulary. We choose SVHN to demonstrate the robustness of V3 in learning content and style disentanglement in a much more challenging setting.

D.1.4 Sprites

The original Sprites dataset, collected from [149] and adopted by [147], contains animated cartoon characters in the pixel graphic style with random appearances and actions. The original Sprites dataset contains animations of six different actions in four perspectives. We collect the Sprites with Actions dataset used in this study by selecting 3 distinct actions in 3 perspectives, resulting in 9 different actions in total, and rendering videos of characters performing actions from these 9 categories randomly, using the critical frames from each action animation. The dataset contains 2160 videos in total, each of which has 9 frames. The characters differ in their hair, body, top, and bottom, forming 2160 unique characters in total. We use 80% of the characters for training and the rest for validation and testing. Examples of the dataset can be viewed on the right of Figure 8.1.

D.1.5 Libri100

The Libri100 dataset is a subset of the LibriSpeech dataset [189], containing the “train-clean-100”, “dev-clean”, and “test-clean” subsets. There are 331 different speakers in total, in which 165 are female and 166 are male. There are no overlapping speakers between the train, validation, and test divisions. Given audio files and ground truth transcriptions, we align the audio with the 39 phonemes used in English using the Montreal Forced Aligner [172]. After normalizing the cropped fragments, we extract the mel spectrograms with a window size of 16ms, hop size of 5ms, and 80 mel bands. The 39 phonemes are indexed as shown in Table D.3.

Index	Phoneme	Index	Phoneme	Index	Phoneme
0	eh	13	ao	26	l
1	z	14	ey	27	k
2	s	15	hh	28	m
3	uw	16	y	29	ch
4	aw	17	f	30	ng
5	oy	18	r	31	t
6	dx	19	g	32	w
7	dh	20	v	33	ae
8	uh	21	ah	34	iy
9	aa	22	er	35	th
10	d	23	ow	36	ay
11	p	24	sh	37	ih
12	n	25	b	38	jh

Table D.3: List of phonemes with their indices.

D.2 Implementation Details

D.2.1 Model Architecture

On InsNotes, we instantiate V3 model using a ResNet18 encoder and a ResNet18T decoder with bottlenecks [93]. In the encoder, the number of channels in the first convolutional layer is set to 64, and gradually increases to 512 in the last layer. The first half of the encoder uses a kernel size of 9, and the second half uses a kernel size of 5. The decoder is symmetric to the encoder. The latent dimension is set to 512. The total number of trainable parameters is 55M.

On PhoneNums and Sprites, we instantiate V3 model using a ResNet encoder and a ResNetT decoder half as deep as the pitch and timbre learning task. Similarly, the number of channels in the first convolutional layer is set to 16, and gradually increases to 256 in the last layer. The encoder uses a kernel size of 5. The decoder is symmetric to the encoder. The latent dimension is set to 512. The total number of trainable parameters is 20M on PhoneNums and 25M on Sprites.

On SVHN, we add one more ResNet layer in every ResBlock on top of the ResNet encoder used in PhoneNums and Sprites. The number of channels in the first convolutional layer is set to 32, and gradually increases to 512 in the last layer. The first half of the encoder uses a kernel size of 5, and the second half uses a kernel size of 3. The decoder is symmetric to the encoder. The latent dimension is set to 768. The total number of trainable parameters is 37M.

On Libri100, we use a similar architecture as InsNotes, but with a maximum number of channels of 256. We deepen the encoder with 2 more ResNet blocks in each layer. The total number of trainable parameters is 24M.

We use the same neural network architecture as V3 for the MINE-based baseline and the cycle loss-based baseline, except that the style branch of the MINE-based method has a variational latent layer. For the MINE-based baseline, we use a 3-layer multi-layer perceptron with 512 hidden units to estimate the mutual information. For the supervised baselines EC^2-VAE (c), we replace the VQ layer of the content branch with a linear layer projecting to the dimension of prediction logits. Besides, the encoder output of the style branch is the mean and log variance vectors instead of representation vectors, which means the style branch is a variational autoencoder (VAE) [125]. For the fully supervised baseline EC^2-VAE (c/s), we project the reparameterized style vectors to the dimension of prediction logits.

D.2.2 Training Details

For all models, we use the Adam optimizer with a learning rate of 0.001 [124]. The fragment sizes on PhoneNums, InsNotes, SVHN, and Sprites are set to 10, 12, 2, and 6, respectively. The relativity r is set to 15, 15, 5, 10 and 5 on PhoneNums, InsNotes, SVHN, Sprites and Libri100, respectively (Generally, we recommend setting a higher r, such as 15, on datasets with clean content and style separation, and setting a lower r, such as 5, on more complex datasets where reconstruction might need to be emphasized more.). The V3 loss weight β is defaultly set to 1 on InsNotes task, and 0.1 in other datasets. For all VQ-based models, we update the codebooks using an exponential moving average with a decay rate of 0.95 [184]. The commitment loss weight α is set to 0.01. On PhoneNums and InsNotes, we set a threshold of $\frac{n}{10K}$ for dead code relaunching to improve codebook utilization, where n is the total number of fragments in a batch. On SVHN, as most images have only 2 or 3 digits, we concatenate fragments in different samples for a higher content coverage in practice to stabilize training. Similarly, on Librispeech, as many consonant phonemes do not exhibit distinct styles like vowels, we also smooth the styles by taking the average of adjacent fragments in practice. For MINE-based baseline models, we update the MINE network once every global iteration using the Adam optimizer and adaptive gradient scaling [15, 232]. The learning rate of the MINE network is set to 0.0002.

We train all models using an exponential decay learning rate scheduler, and take the model with the best validation loss as the final model. All models are trained on a single Nvidia RTX 4090 GPU. The V3 loss should decay to zero within a few epochs after training starts. All supervised learning methods converge within 2 hours, while the converging time of all unsupervised learning methods differs from 5 hours to 24 hours.

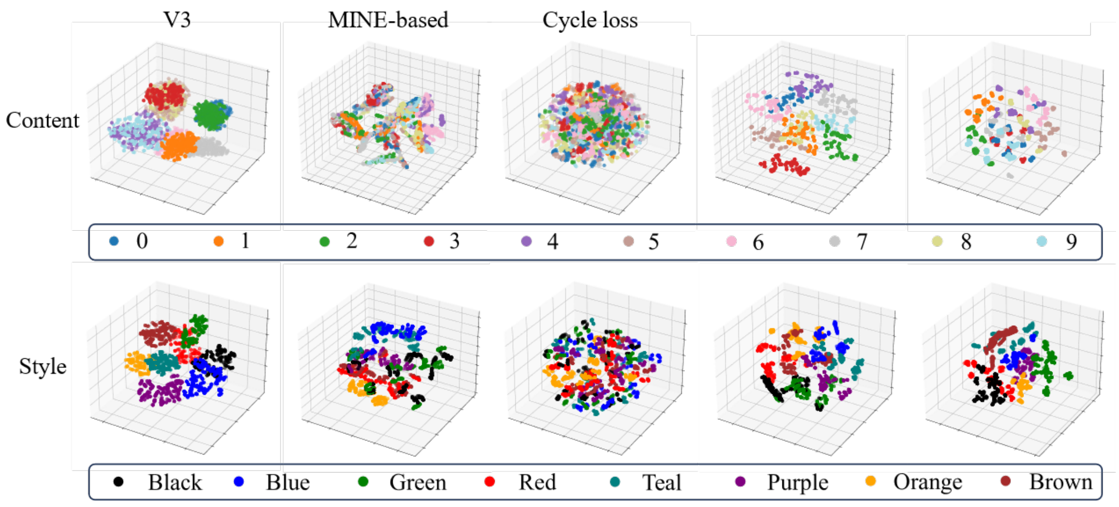


Figure D.3: t-SNE visualization of the learned digit (content) and color (style) representations on PhoneNums when there is no codebook redundancy ($K = 10$).

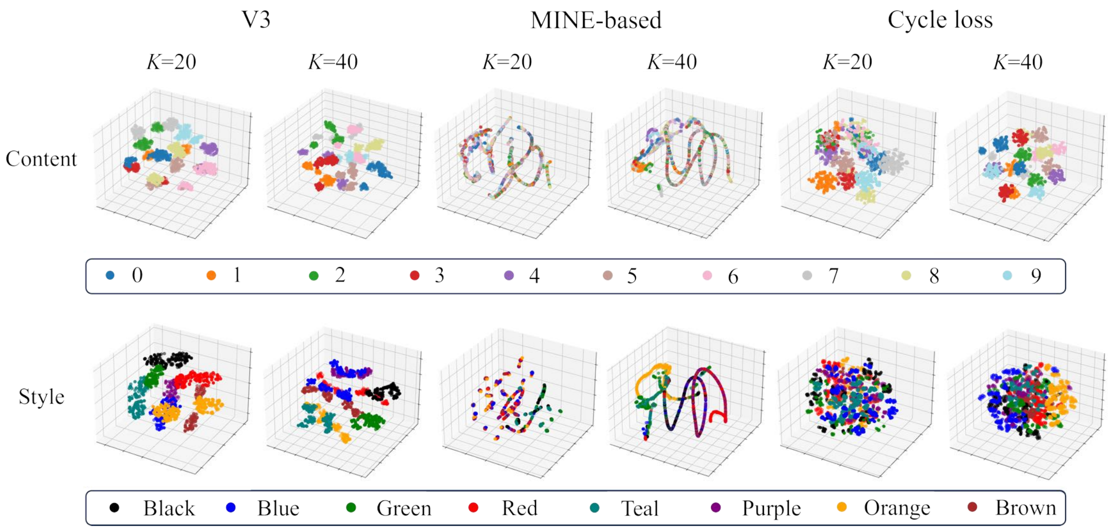


Figure D.4: t-SNE visualization of the learned digit (content) and color (style) representations on PhoneNums when the codebooks are redundant.

D.3 More Experiment Results

In this part, we provide extra experiment results in addition to the results in Section 8.3. We will focus more on the visualizations of the learned content and style representations, and the alignment between the learned codebooks and the ground truth content labels.

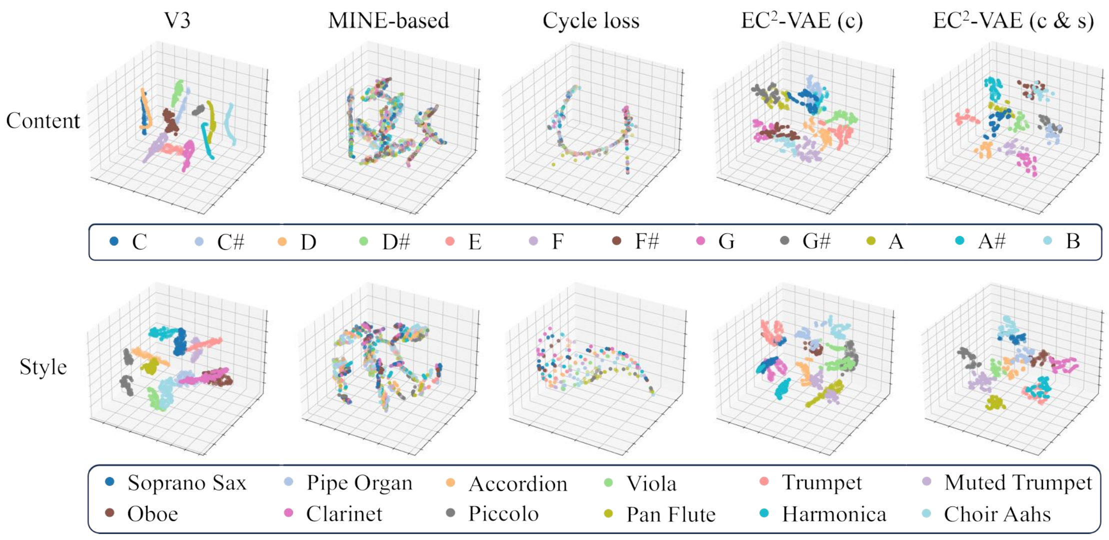


Figure D.5: t-SNE visualization of the learned pitch (content) and timbre (style) representations on InsNotes when there is no codebook redundancy ($K = 12$).

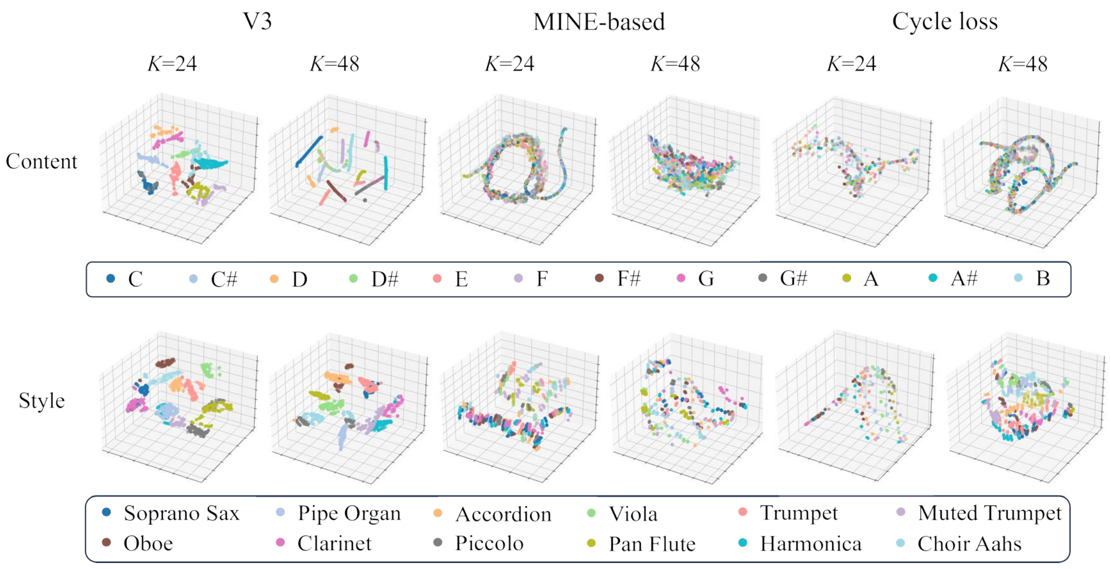


Figure D.6: t-SNE visualization of the learned pitch (content) and timbre (style) representations on InsNotes when the codebooks are redundant.

D.3.1 Results of Content-Style Disentanglement

This section provides 3-dimensional t-SNE visualization results of the learning content and style representations in support of Section 8.3.2. We show that on different datasets and under

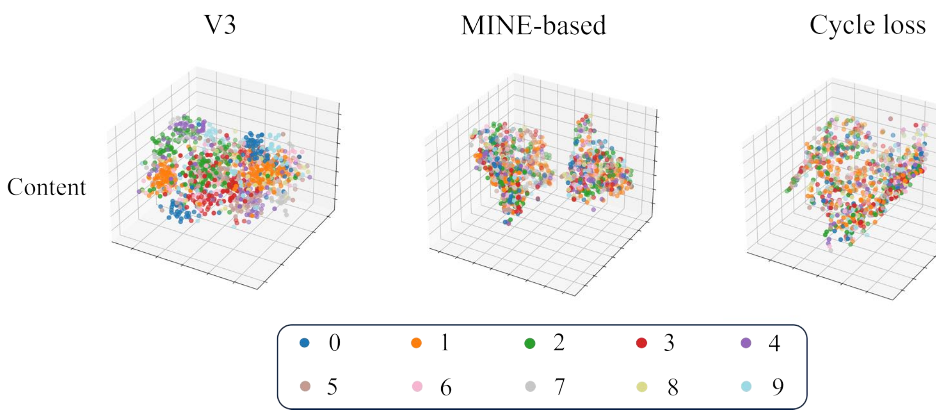


Figure D.7: t-SNE visualization of the learned content (digit) representations on SVHN.

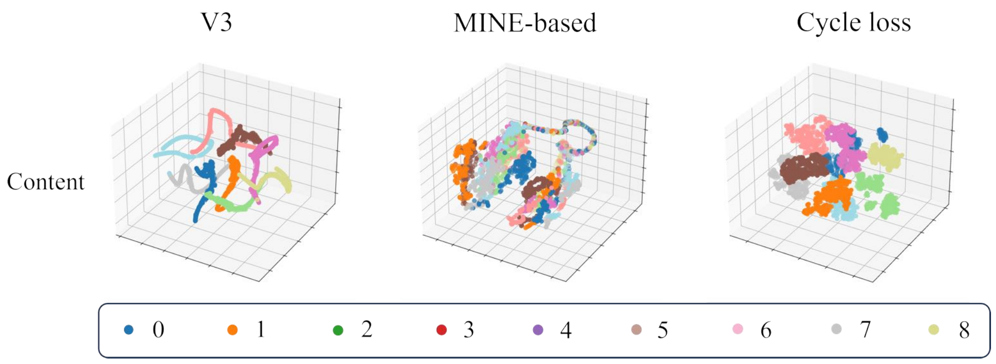


Figure D.8: t-SNE visualization of the learned content (action) representations on Sprites. The 10 colors refer to 10 different actions.

different K settings, content and style representations learned by V3 show the clearest groupings compared to baselines, and the groupings match well with ground truth content and style labels.

On PhoneNums, we first visualize with t-SNE the learned content and style representations when there is no codebook redundancy ($K = 10$), and color them by the ground truth content or style labels. We set $K = 12$ for learning digits and colors. The results are shown in Figure D.3. We can see that V3 learns clearer content and style representations in groups compared to unsupervised baselines. When the codebooks contain redundancy, the results are shown in Figure D.4. We can see that V3 still achieves the clearest content and style grouping.

On InsNotes, the visualizations of $\boldsymbol{z}^{\mathrm{c}}$ and $\boldsymbol{z}^{\mathrm{s}}$ when $K = 12$ are shown in Figure D.5, and the visualizations when the codebook is redundant are shown in Figure D.6. Both results also show

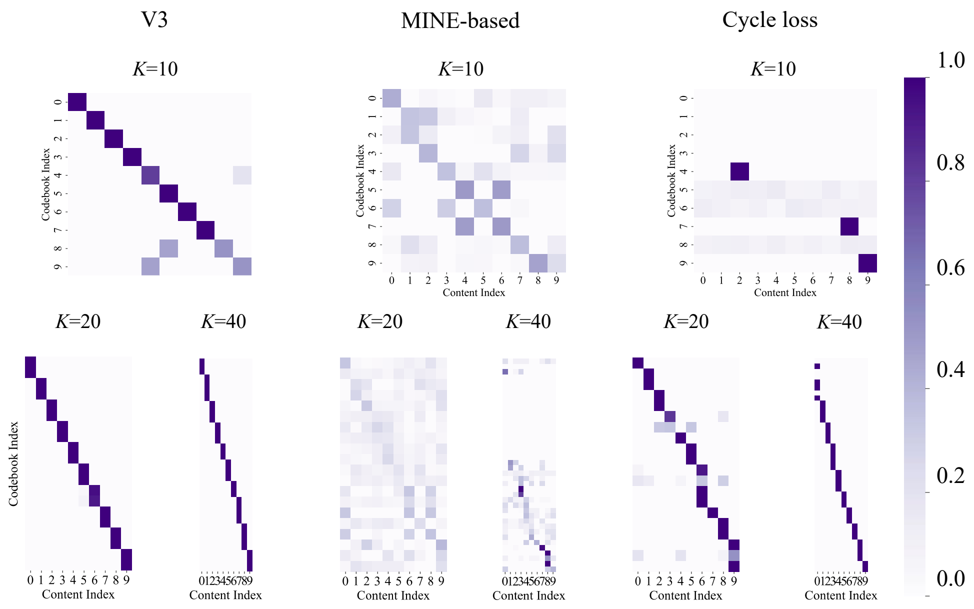


Figure D.9: Confusion matrices of learned codebooks on PhoneNums. The horizontal axes show digit labels from "0" to "9", and the vertical axes show codebook atoms sorted by ground truth digit labels.

V3 groups content and style better than baselines.

On SVHN, the visualizations of $\boldsymbol{z}^{\mathrm{c}}$ are shown in Figure D.7. Since SVHN does not have discrete style labels, we only show the grouping of content representations. V3 is trained at $K = 20$. Although not as good as the results shown in Figure D.3 on PhoneNums, V3 still achieves the best grouping of digits with learned content representations among unsupervised methods.

On Sprites, there is no discrete style label either. Figure D.8 shows the t-SNE visualizations of $\boldsymbol{z}^{\mathrm{c}}$, and V3 is trained at $K = 18$. Both V3 and Cycle loss achieve good content grouping, but it is observable that some clusters of the cycle loss $\boldsymbol{z}^{\mathrm{c}}$ have broken into several subclusters, indicating that there is still content and style entanglement. This is also supported by Section 8.3.2.

D.3.2 Results of Symbolic Content Interpretability

This section provides intuitive visualizations about how the learned content codebook entries align with ground truth content labels in Section 8.3.4. We first collect frequencies of every content encoded to every codebook entry, and then permute the codebook to make the confusion

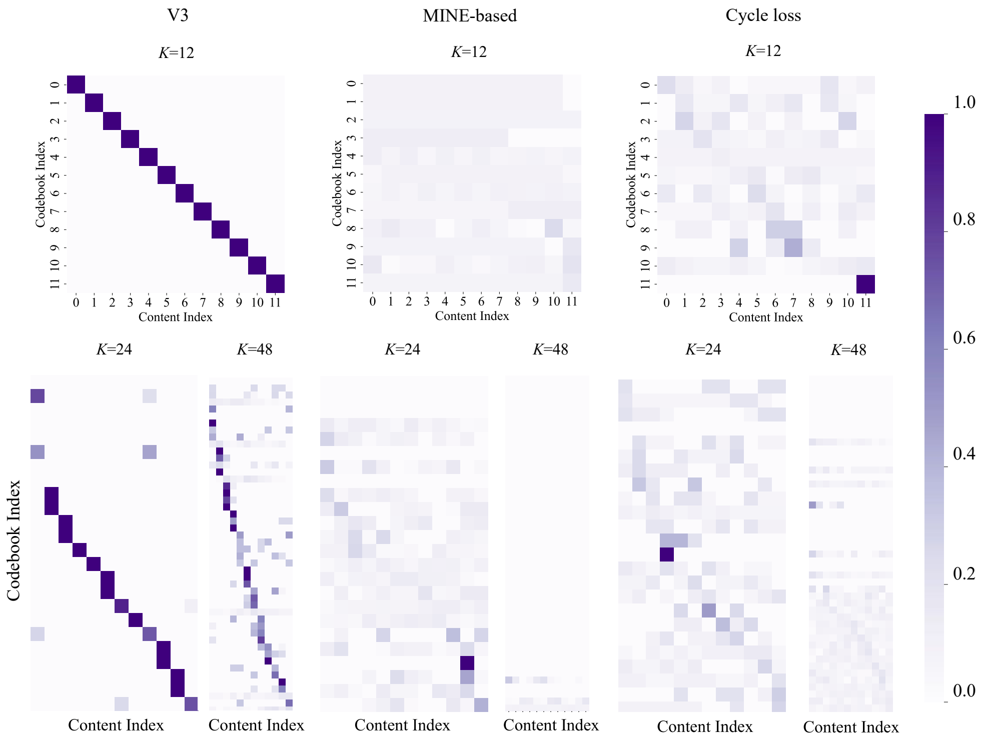


Figure D.10: Confusion matrices of learned codebooks on InsNotes. The horizontal axes show pitch labels from "C" to "B", and the vertical axes show codebook atoms sorted by ground truth pitch labels.

matrix look like an eye for a clear alignment. Then we plot heatmaps of confusion matrices between codebook entries (vertical axes) and content labels (horizontal axes).

On PhoneNums and InsNotes, we plot the confusion matrices under different K settings in Figure D.9 and Figure D.10, respectively. The results show V3 achieves the clearest symbol interpretability in all K settings. Results on SVHN and Sprites are shown in Figure D.11 and Figure D.12. On Sprites, both V3 and cycle loss learns codebooks with good interpretability, but V3 still has fewer misclassifications. Although V3 does not learn a clear one-to-one codebook entry to content label mapping on SVHN, it still shows a clearer alignment relationship than other methods. An interesting fact is the order of learning we observe during training—V3 usually first distinguish 0 and 1, then start to understand 2 is different, then 3. It often confuses between 5 and 6 and between 1 and 7, and it usually fails to learn 8 and 9. This human-like learning trajectory might be subject to both the ratio of content classes and their pairwise similarities in shape. The

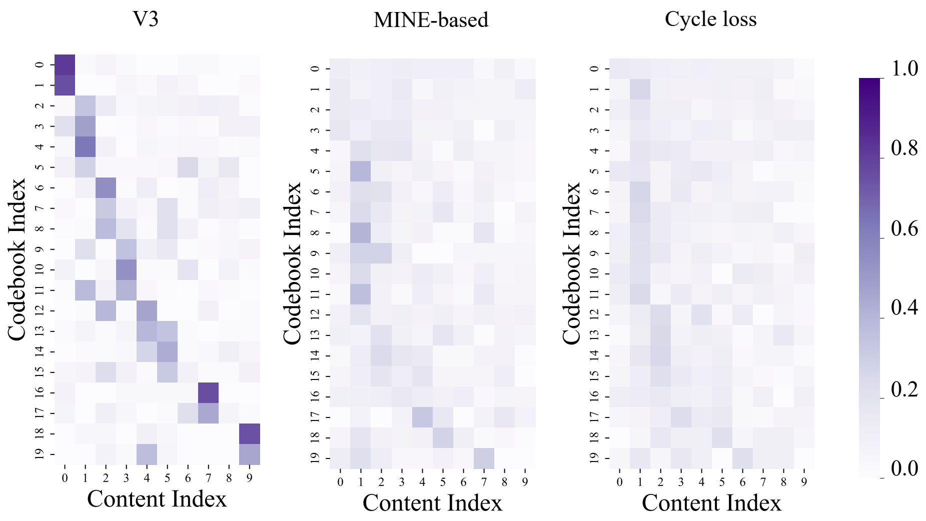


Figure D.11: Confusion matrices of learned codebooks on SVHN. The horizontal axes show digit labels from "0" to "9", and the vertical axes show codebook atoms sorted by ground truth digit labels.

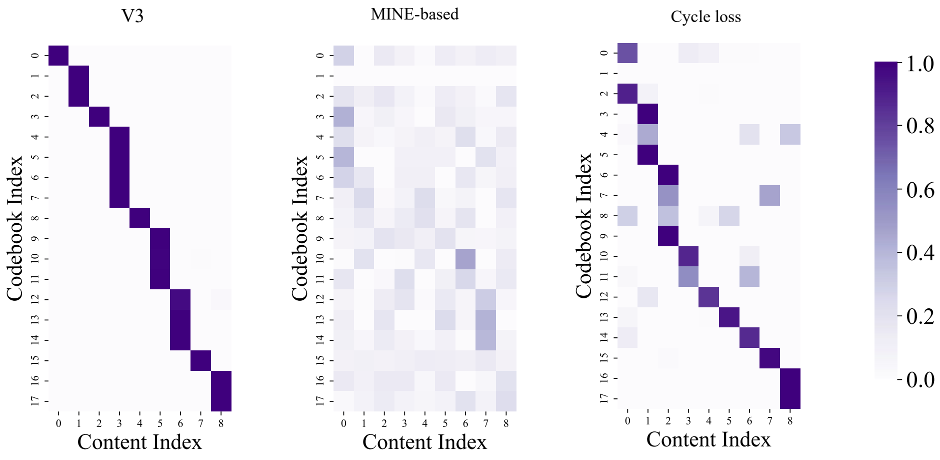


Figure D.12: Confusion matrices of learned codebooks on Sprites. The horizontal axes are different action labels, and the vertical axes show codebook atoms sorted by ground truth action labels.

similar phenomenon is observed in the confusion matrices on Libri100 as shown in Figure D.13. V3 distinguishes between consonants and vowels pretty well, and confuses between "z" and "s", and between "n" and "ng", which are phonetically similar.

To further investigate the disentanglement ability of models, we perform latent representation recombination using the trained models. Figure 8.3 has already demonstrated the results on SVHN of decoding a fixed z^{s} with every z^{c}. Here we show the results on PhoneNums, where instead of

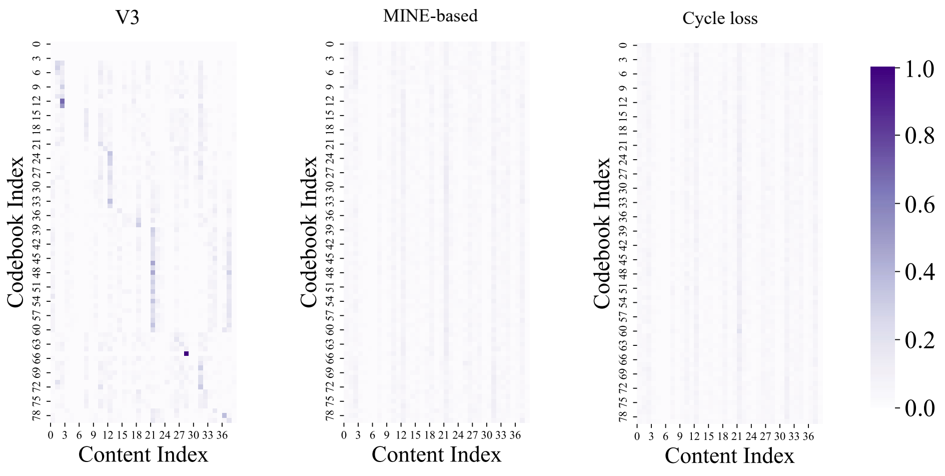


Figure D.13: Confusion matrices of learned codebooks on Libri100. The horizontal axes are different phoneme labels, and the vertical axes show codebook atoms sorted by ground truth phoneme labels.

using a fixed $\boldsymbol{z}^{\mathrm{s}}$ encoded from an example, we compute the mean $\boldsymbol{z}^{\mathrm{s}}$ of all fragments from a class as its style representations for decoding. We select the V3 model with $K = 10$, align codebook entries with digit labels, and enumerate all combinations of $\boldsymbol{z}^{\mathrm{c}}$ and $\boldsymbol{z}^{\mathrm{s}}$. We present the results in Figure D.14. Compared to baselines, V3 can fairly well reconstruct the involved 8 colors and the digits from 0 to 9, even though it is not informed with any discrete labels during training. In contrast, the MINE-based baseline and the cycle loss baseline fail to distinguish the digits, although the color reconstruction is not bad. They generate blurry digits that look like "5", "8" or "6", which are the most conservative choices. As for results on the music dataset InsNotes, we refer you to our web demo page for an interactive experience.

D.4 Ablation Study

For ablation, we experiment with another type of variability measurement $\nu_k(\cdot)$, which is standard deviation (SD). Besides, we train four variants of V3, each without one of the four regularization terms defined in Equation 8.9-8.12. We conduct experiments on PhoneNums and InsNotes, two datasets with style labels available, and evaluate the content and style disentan-

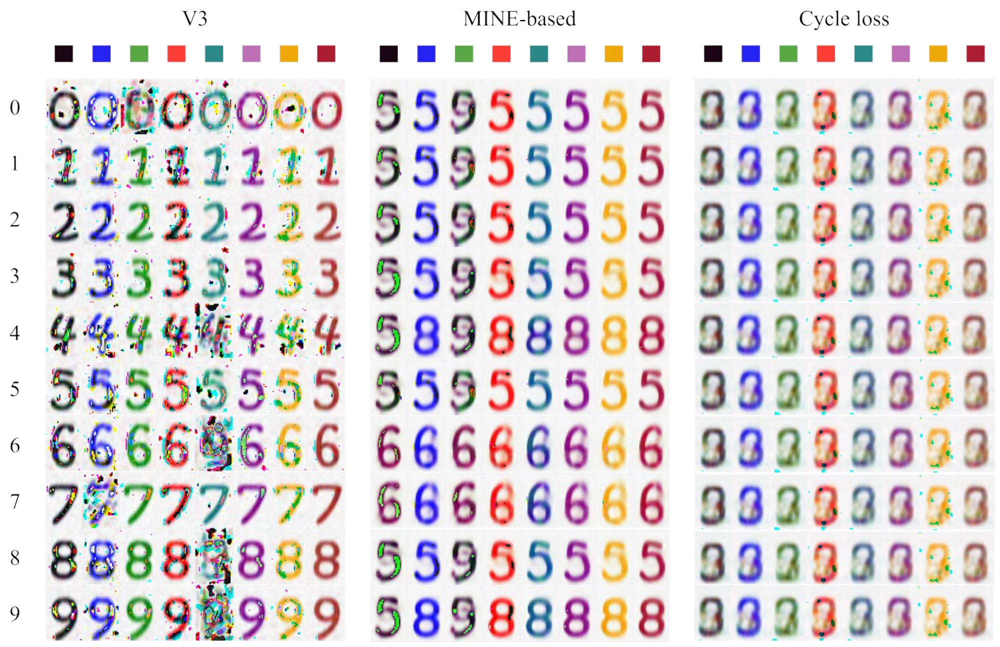


Figure D.14: Comparison of generated digits by recombining content and style latents using unsupervised methods trained on PhoneNums.

glement performances. The results are reported in Table D.4 and Table D.5. It can be seen that $\nu_k = \text{SD}$ does not work as well as $\nu_k = \text{MPD}$, which can be explained by its weakness in constraining multi-peak content distributions within samples. It is also worth noting that V3 sometimes performs fairly well even when discarding one of its terms. In these cases, we observe a decrease in the discarded loss even if we do not explicitly optimize for it. We suspect this is due to the robustness of V3 constraints as reflected in the symmetric relationships among the four losses—we can enforce three relations, and the fourth one may fall into the right place automatically. However, in practice, it is difficult to tell the one term to free beforehand as it is also related to detailed content and style variations in specific domains. As a result, the V3 constraints as a whole show robust performance across domains.

Method	K	Content				Style			
		PR-AUC		Best F1		PR-AUC		Best F1	
		$\boldsymbol{z}^{\mathrm{c}} \uparrow$	$\boldsymbol{z}^{\mathrm{s}} \downarrow$	$\boldsymbol{z}^{\mathrm{c}} \uparrow$	$\boldsymbol{z}^{\mathrm{s}} \downarrow$	$\boldsymbol{z}^{\mathrm{c}} \downarrow$	$\boldsymbol{z}^{\mathrm{s}} \uparrow$	$\boldsymbol{z}^{\mathrm{c}} \downarrow$	$\boldsymbol{z}^{\mathrm{s}} \uparrow$
V3	10	83.2	12.8	84.1	18.5	**14.9**	**95.4**	22.6	**91.0**
V3 ($\nu_k = \mathrm{SD}$)	10	42.7	17.9	53.5	21.5	18.2	49.9	24.7	51.6
V3 (w/o $\mathcal{L}_{\text{content}}$)	10	43.8	13.8	51.2	18.9	18.1	90.8	**22.4**	87.5
V3 (w/o $\mathcal{L}_{\text{style}}$)	10	64.6	13.2	70.3	18.7	17.7	87.8	24.5	83.9
V3 (w/o $\mathcal{L}_{\text{fragment}}$)	10	**96.3**	**11.7**	**98.9**	**17.9**	15.9	90.1	23.2	88.5
V3 (w/o $\mathcal{L}_{\text{sample}}$)	10	47.0	12.7	57.8	18.9	15.4	88.4	24.7	85.3

Table D.4: Ablation study of V3 settings on content-style disentanglement performance on PhoneNums. Values are reported in percentage.

Method	K	Content				Style			
		PR-AUC		Best F1		PR-AUC		Best F1	
		$\boldsymbol{z}^{\mathrm{c}} \uparrow$	$vz^{\mathrm{s}} \downarrow$	$\boldsymbol{z}^{\mathrm{c}} \uparrow$	$\boldsymbol{z}^{\mathrm{s}} \downarrow$	$\boldsymbol{z}^{\mathrm{c}} \downarrow$	$\boldsymbol{z}^{\mathrm{s}} \uparrow$	$\boldsymbol{z}^{\mathrm{c}} \downarrow$	$\boldsymbol{z}^{\mathrm{s}} \uparrow$
V3	12	**89.9**	8.9	**90.1**	15.1	**9.3**	**87.5**	**15.0**	**88.0**
V3 ($\nu_k = \mathrm{SD}$)	12	12.9	9.9	17.5	15.0	16.3	24.7	24.1	36.5
V3 (w/o $\mathcal{L}_{\text{content}}$)	12	19.2	9.3	28.0	14.3	13.6	66.2	19.0	68.4
V3 (w/o $\mathcal{L}_{\text{style}}$)	12	72.1	8.9	14.2	84.0	13.7	78.7	23.6	79.0
V3 (w/o $\mathcal{L}_{\text{fragment}}$)	12	26.0	12.1	35.7	17.6	13.5	53.7	20.1	56.7
V3 (w/o $\mathcal{L}_{\text{sample}}$)	12	86.4	**7.9**	89.3	**14.2**	11.3	50.7	19.4	56.2

Table D.5: Ablation study of V3 settings on content-style disentanglement performance on InsNotes. Values are reported in percentage.

D.5 Discussion

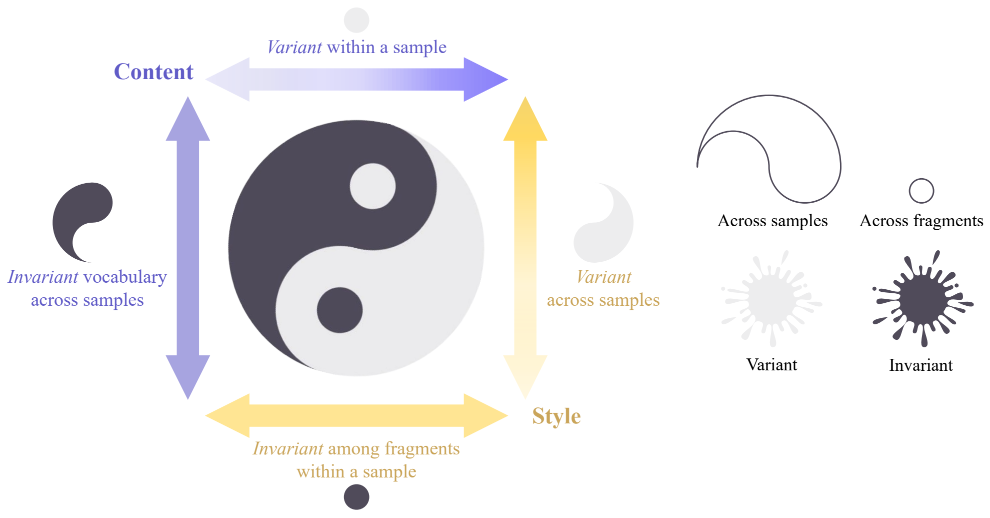
Connection between Content-Style Disentanglement and OOD Generalizability: Disentanglement can intuitively boost OOD generalization for several key reasons. By separating different factors, like content and style, the model can focus on the important features without getting distracted by irrelevant variations. This separation makes the model more robust to changes. For instance, if the style changes in an OOD sample while the content remains similar, the model might still recognize and process the content effectively. Additionally, disentangled representations often lead to more generalized features, enabling the model to identify important patterns that are invariant across different distributions. This property facilitates easier transfer learning because models with disentangled representations can be more readily fine-tuned for new tasks, as supported by our experiments in Section 8.3.3.

Connection between Content-Style Disentanglement and Symbolic Interpretability: In Section 8.3.2 and Section 8.3.4, we separately examined content-style disentanglement and symbolic-level interpretability. This discussion now seeks to understand how these elements are interconnected—specifically, whether V3's disentangled representation space inherently improves symbolic-level interpretability.

The transition from purely observational data to symbolic representation remains an open question in cognitive science and artificial intelligence. We suggest that robust content-style disentanglement is closely linked to better symbolic interpretability, as evidenced in Tables 8.1 to 8.4, 8.6 and 8.7. These figures show that both V3 and supervised methods, which achieve better disentanglement, also provide superior interpretability compared to methods less effective in disentanglement (supervised classification—while it is not—can here be viewed as another form of interpretable VQ symbols). Additionally, as illustrated in Figures D.3 to D.8 , well-disentangled style spaces (meaning they contain less content information) see well-formed clusters, which can facilitate straightforward postprocessing for discrete and symbolic labeling.

Connection between Content-Style Disentanglement and Philosophy: The statistical patterns and mutual relationship between content and style closely correspond to the philosophical concept of Yin and Yang, a fundamental duality in universal balance and dynamics. In the famous Yin-Yang diagram, Yin and Yang are equally divided and composed of two identical shapes (the fish-like swirl and the small dot) with opposing colors (light and dark), together forming the completeness of the world [258].

We can interpret the two fundamental shapes as the two axes along which data is observed—the swirl represents observation across samples, while the dot represents observation across fragments within samples. The two colors signify two kinds of dynamics—light denotes variant, and



Variant within a sample
Content
Invariant vocabulary across samples
Variant across samples
Style
Invariant among fragments within a sample
Across samples
Across fragments
Variant
Invariant

Figure D.15: An illustration of the correspondence between the content-style duality and the Yin-Yang duality.

dark denotes invariant. As a result, the duality of Yin and Yang becomes the duality of content and style. Content shows variability within samples (the light dot) and invariability of vocabulary across samples (the dark swirl), while style shows variability across samples (the light swirl) and invariability within a sample (the dark dot). They can be disentangled from data as two components, yet neither can exist alone.

V3 and related work: We explicate the difference and connection between V3 and several other most relevant works as follows.

- InfoGAN [39]: InfoGAN is similar to our approach in that both models learn interpretable representations and decouple these representations from the data. However, there are several key differences: 1) Each representation in InfoGAN is of very low dimensionality; 3) The specific aspects learned are less controllable, while V3 focuses on learning the distinctions of content and style; 3) GAN is known to be less unstable in training than autoencoders and VAEs, and it is a framework more for generative modeling than representation learning.

- DSAE and variants [9, 103, 147, 163, 164]: DSAE shares similar insights with V3 regarding the intrinsic relationship between content and style, but it primarily focuses on the invariability of style and the variability of content within a sample, giving less attention to the invariability of content vocabulary and the variability of style across a broader scope. Other distinctions include: 1) DSAE focuses on learning a fixed style representation for a whole sample, which may struggle with samples where the style varies, such as singing performances that feature both chest voice and falsetto, or instrument performances with multiple articulations; 2) The DSAE family requires access to the entire sequence when encoding style; 3) Most importantly, the content learned in DSAE is context-dependent, while V3 emphasizes on learning more universal content representations.

- VICReg [11]: V3 has a similar form of loss function as VICReg, and both models leverage variance and invariance among entities to help train representation learning frameworks. However, VICReg focuses on learning a single representation for each entity, while V3 focuses on learning disentangled representations. Also, VICReg learns discriminative representations without clear interpretability, but V3 learns interpretable content symbols and has a decoding ability to recombine content-style pairs. In fact, we draw on their mathematical representations and the idea of using regularization to prevent latent representation from collapsing, a concept also advocated by [140].

Bibliography

[1] ABC Wiki. *The abc Notation System*. 2021. URL: https://abcwiki.org/abc:syntax.

[2] ADSR. *ADSR Sample Manager*. URL: https://www.adsrsounds.com/product/software/adsr-sample-manager/.

[3] Andrea Agostinelli et al. "MusicLM: Generating Music From Text". In: *CoRR* abs/2301.11325 (2023).

[4] Sven Ahlbäck. "Melody beyond notes: A study of melody cognition". PhD thesis. Göteborgs universitet, 2004.

[5] Taketo Akama. "Controlling Symbolic Music Generation based on Concept Learning from Domain Knowledge". In: *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*. 2019, pp. 816–823.

[6] Jean-Baptiste Alayrac et al. "Flamingo: a Visual Language Model for Few-Shot Learning". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*. 2022.

[7] Zeyuan Allen-Zhu and Yuanzhi Li. "Physics of Language Models: Part 3.2, Knowledge Manipulation". In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[8] Shunya Ariga, Satoru Fukayama, and Masataka Goto. “Song2Guitar: A Difficulty-Aware Arrangement System for Generating Guitar Solo Covers from Polyphonic Audio of Popular Music”. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*. 2017, pp. 568–574.

[9] Junwen Bai, Weiran Wang, and Carla P. Gomes. “Contrastively Disentangled Sequential Variational Autoencoder”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 2021, pp. 10105–10118.

[10] Hayeon Bang et al. “PIAST: A Multimodal Piano Dataset with Audio, Symbolic and Text”. In: *CoRR* abs/2411.02551 (2024).

[11] Adrien Bardes, Jean Ponce, and Yann LeCun. “VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[12] Lawrence W Barsalou. “Perceptual Symbol Systems”. In: *Behavioral and brain sciences* 22.4 (1999), pp. 577–660.

[13] Peter W. Battaglia et al. “Relational inductive biases, deep learning, and graph networks”. In: *CoRR* abs/1806.01261 (2018).

[14] Gulcin Baykal et al. “ProtoDiffusion: Classifier-Free Diffusion Guidance with Prototype Learning”. In: *CoRR* abs/2307.01924 (2023).

[15] Mohamed Ishmael Belghazi et al. “Mutual information neural estimation”. In: *International conference on machine learning*. PMLR. 2018, pp. 531–540.

[16] Emmanouil Benetos et al. “Automatic Music Transcription: An Overview”. In: *IEEE Signal Process. Mag.* 36.1 (2019), pp. 20–30.

[17] Emmanouil Benetos et al. “Automatic music transcription: challenges and future directions”. In: *J. Intell. Inf. Syst.* 41.3 (2013), pp. 407–434.

[18] Keshav Bhandari et al. “Text2midi: Generating Symbolic Music from Captions”. In: *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. AAAI Press, 2025, pp. 23478–23486.

[19] Pieter-Tjerk de Boer et al. “A Tutorial on the Cross-Entropy Method”. In: *Ann. Oper. Res.* 134.1 (2005), pp. 19–67.

[20] Russell Sammut Bonnici, Martin Benning, and Charalampos Saitis. “Timbre Transfer with Variational Auto Encoding and Cycle-Consistent Adversarial Networks”. In: *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*. IEEE, 2022, pp. 1–8.

[21] Ali Borji. “Qualitative failures of image generation models and their application in detecting deepfakes”. In: *Image Vis. Comput.* 137 (2023), p. 104771.

[22] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. “Multi-Level Variational Autoencoder: Learning Disentangled Representations From Grouped Observations”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, 2018, pp. 2095–2102.

[23] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. “Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription”. In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[24] Konstantinos Bousmalis et al. “Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 95–104.

[25] Samuel R. Bowman et al. “Generating Sentences from a Continuous Space”. In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*. ACL, 2016, pp. 10–21.

[26] Jean-Pierre Briot. “From artificial neural networks to deep learning for music generation: history, concepts and trends”. In: *Neural Comput. Appl.* 33.1 (2021), pp. 39–65.

[27] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. “Deep Learning Techniques for Music Generation - A Survey”. In: *CoRR* abs/1709.01620 (2017).

[28] Jean-Pierre Briot and François Pachet. “Deep learning for music generation: challenges and directions”. In: *Neural Computing and Applications* 32.4 (2020), pp. 981–993.

[29] Gino Brunner et al. “JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs”. In: *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017*. IEEE Computer Society, 2017, pp. 519–526.

[30] Gino Brunner et al. “MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer”. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*. 2018, pp. 747–754.

[31] Allen Clayton Cadwallader and David Gagné. *Analysis of tonal music: A Schenkerian approach*. Oxford University Press, 1998.

[32] Shan Carter and Michael Nielsen. “Using Artificial Intelligence to Augment Human Intelligence”. In: *Distill* 2.12 (2017), e9.

[33] Rodrigo Castellon, Chris Donahue, and Percy Liang. “Codified audio language modeling learns useful representations for music information retrieval”. In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021*. 2021, pp. 88–96.

[34] Haonan Chen et al. “SymPAC: Scalable Symbolic Music Generation With Prompts and Constraints”. In: *Proceedings of the 25th International Society for Music Information Retrieval Conference, ISMIR 2024, San Francisco, California, USA and Online, November 10-14, 2024*. 2024, pp. 1029–1036.

[35] Ke Chen, Gus Xia, and Shlomo Dubnov. “Continuous Melody Generation via Disentangled Short-Term Representations and Structural Conditions”. In: *IEEE 14th International Conference on Semantic Computing, ICSC 2020, San Diego, CA, USA, February 3-5, 2020*. IEEE, 2020, pp. 128–135.

[36] Ke Chen et al. “Music SketchNet: Controllable Music Generation via Factorized Representations of Pitch and Rhythm”. In: *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020, Montreal, Canada, October 11-16, 2020*. 2020, pp. 77–84.

[37] Ke Chen et al. “The Effect of Explicit Structure Encoding of Deep Neural Networks for Symbolic Music Generation”. In: *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*. IEEE, Jan. 2019, pp. 77–84.

[38] Tian Qi Chen et al. “Isolating Sources of Disentanglement in Variational Autoencoders”. In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. 2018, pp. 2615–2625.

[39] Xi Chen et al. “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*

29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. 2016, pp. 2172–2180.

[40] Shin-I Cheng et al. “Adaptively-Realistic Image Generation from Stroke and Sketch with Diffusion Model”. In: *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2023, Waikoloa, HI, USA, January 2-7, 2023.* IEEE, 2023, pp. 4043–4051.

[41] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL.* ACL, 2014, pp. 1724–1734.

[42] Jongho Choi and Kyogu Lee. “Pop2Piano : Pop Audio-Based Piano Cover Generation”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023.* IEEE, 2023, pp. 1–5.

[43] Keunwoo Choi and Kyunghyun Cho. “Deep Unsupervised Drum Transcription”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019.* 2019, pp. 183–191.

[44] Yunjey Choi et al. “StarGAN v2: Diverse Image Synthesis for Multiple Domains”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020.* Computer Vision Foundation / IEEE, 2020, pp. 8185–8194.

[45] Yunjey Choi et al. “StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation”. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018.* Computer Vision Foundation / IEEE Computer Society, 2018, pp. 8789–8797.

[46] Hyeshin Chu et al. “An Empirical Study on How People Perceive AI-generated Music”. In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022.* ACM, 2022, pp. 304–314.

[47] Junyoung Chung et al. “A Recurrent Latent Variable Model for Sequential Data”. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada.* 2015, pp. 2980–2988.

[48] Junyoung Chung et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *CoRR* abs/1412.3555 (2014).

[49] Muzio Clementi, Carl Tausig, and Karl Friedrich Weitzmann. *Gradus ad parnassum.* Peters, 2010.

[50] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences.* 2nd. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

[51] David Cope. “Recombinant Music: Using the Computer to Explore Musical Style”. In: *Computer* 24.7 (1991), pp. 22–28.

[52] Jade Copet et al. “Simple and Controllable Music Generation”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.* 2023.

[53] Bas Cornelissen, Willem H. Zuidema, and John Ashley Burgoyne. “Cosine Contours: a Multipurpose Representation for Melodies”. In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021.* 2021, pp. 135–142.

[54] Shuqi Dai, Huiran Yu, and Roger B. Dannenberg. “What is missing in deep music generation? A study of repetition and structure in popular music”. In: *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022, Bengaluru, India, December 4-8, 2022.* 2022, pp. 659–666.

[55] Shuqi Dai, Huan Zhang, and Roger B Dannenberg. “Automatic analysis and influence of hierarchical structure on melody, rhythm and harmony in popular music”. In: *Proceedings of the 2020 Joint Conference on AI Music Creativity (CSMC-MuMe)*. 2020.

[56] Shuqi Dai, Zheng Zhang, and Gus Xia. “Music Style Transfer Issues: A Position Paper”. In: *CoRR* abs/1803.06841 (2018).

[57] Shuqi 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, Online, November 7-12, 2021*. 2021, pp. 143–150.

[58] Prafulla Dhariwal et al. “Jukebox: A Generative Model for Music”. In: *CoRR* abs/2005.00341 (2020).

[59] Prafulla Dhariwal et al. “Jukebox: A Generative Model for Music”. In: *CoRR* abs/2005.00341 (2020).

[60] Christian Dittmar, Martin Pfleiderer, and Meinard Müller. “Automated Estimation of Ride Cymbal Swing Ratios in Jazz Recordings”. In: *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*. 2015, pp. 271–277.

[61] Carl Doersch. “Tutorial on Variational Autoencoders”. In: *CoRR* abs/1606.05908 (2016).

[62] Chris Donahue and Percy Liang. “Sheet sage: Lead sheets from music audio”. In: *Proc. ISMIR Late-Breaking and Demo* (2021).

[63] Chris Donahue, John Thickstun, and Percy Liang. “Melody transcription via generative pre-training”. In: *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022, Bengaluru, India, December 4-8, 2022*. 2022, pp. 485–492.

[64] Chris Donahue et al. “LakhNES: Improving Multi-instrumental Music Generation with Cross-domain Pre-training”. In: *Proceedings of the 20th International Society for Music In-*

formation Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019. 2019, pp. 685–692.

[65] Hao-Wen Dong et al. "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018.* AAAI Press, 2018, pp. 34–41.

[66] Chris Dyer et al. "Recurrent Neural Network Grammars". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* San Diego, California: Association for Computational Linguistics, June 2016, pp. 199–209.

[67] Anders Elowsson and Anders Friberg. "Algorithmic composition of popular music". In: *Proceedings of the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music.* 2012, pp. 276–285.

[68] Jesse H. Engel et al. "DDSP: Differentiable Digital Signal Processing". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net, 2020.

[69] Philippe Esling, Axel Chemla-Romeu-Santos, and Adrien Bitton. "Bridging Audio Analysis, Perception and Synthesis with Perceptually-regularized Variational Timbre Spaces". In: *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018.* 2018, pp. 175–181.

[70] Christoph Finkensiep and Martin Rohrmeier. “Modeling and Inferring Proto-Voice Structure in Free Polyphony”. In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021.* 2021, pp. 189–196.

[71] S Forsgren and H Martiros. “Riffusion-Stable diffusion for real-time music generation”. In: *URL https://riffusion.com/about* (2022).

[72] Eric Foxley. *Nottingham database.* 2011.

[73] Yang Gao, Rita Singh, and Bhiksha Raj. “Voice Impersonation Using Generative Adversarial Networks”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018.* IEEE, 2018, pp. 2506–2510.

[74] Yuan Gao. “Towards neural music style transfer”. New York University, 2017.

[75] Hugo Flores García et al. “VampNet: Music Generation via Masked Acoustic Token Modeling”. In: *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023, Milan, Italy, November 5-9, 2023.* 2023, pp. 359–366.

[76] Josh Gardner et al. “MT3: Multi-Task Multitrack Music Transcription”. In: *The Tenth International Conference on Learning Representations, ICLR 2022.* OpenReview.net, 2022.

[77] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. “A Neural Algorithm of Artistic Style”. In: *CoRR* abs/1508.06576 (2015).

[78] Ian J. Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada.* 2014, pp. 2672–2680.

[79] Fabien Gouyon et al. “An experimental comparison of audio tempo induction algorithms”. In: *IEEE Trans. Speech Audio Process.* 14.5 (2006), pp. 1832–1844.

[80] Jean-Bastien Grill et al. "Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* 2020.

[81] Ryan Groves. "Automatic Melodic Reduction Using a Supervised Probabilistic Context-Free Grammar". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016.* 2016, pp. 775–781.

[82] Xiangming Gu et al. "Automatic Lyric Transcription and Automatic Music Transcription from Multimodal Singing". In: *ACM Trans. Multim. Comput. Commun. Appl.* 20.7 (2024), 209:1–209:29.

[83] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. "DeepBach: a Steerable Model for Bach Chorales Generation". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017.* Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1362–1371.

[84] Danijar Hafner et al. "Mastering Diverse Domains through World Models". In: *CoRR* abs/2301.04104 (2023).

[85] Hans Hagen. "Advanced animation and rendering techniques : A Watt Addison-Wesley, UK (1992) 840 pp, ISBN 0 201 544 121". In: *Comput. Aided Des.* 26.2 (1994), p. 157.

[86] Masatoshi Hamanaka. *GTTM Database.* https://gttm.jp/gttm/database/. 2009.

[87] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. "σGTTM III: Learning-Based Time-Span Tree Generator Based on PCFG". In: *Music, Mind, and Embodiment - 11th International Symposium, CMMR 2015, Plymouth, UK, June 16-19, 2015, Revised Selected Papers.* Vol. 9617. Lecture Notes in Computer Science. 2015, pp. 387–404.

[88] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. “deepGTTM-II: Automatic Generation of Metrical Structure Based on Deep Learning Technique”. In: *Proceedings of the 13th Sound and Music Conference*. 2016, pp. 221–249.

[89] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. “Implementing ‘A generative theory of tonal music’”. In: *Journal of New Music Research* 35.4 (2006), pp. 249–277.

[90] Curtis Hawthorne et al. “Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[91] Curtis Hawthorne et al. “Onsets and Frames: Dual-Objective Piano Transcription”. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*. 2018, pp. 50–57.

[92] Curtis Hawthorne et al. “Transformer-nade for piano performances”. In: *NIPS 2nd workshop on machine learning for creativity and design*. 2018.

[93] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778.

[94] Qi He, Gus Xia, and Ziyu Wang. “TOMI: Transforming and Organizing Music Ideas for Multi-Track Compositions with Full-Song Structure”. In: *Proceedings of the 26th International Society for Music Information Retrieval Conference*. Accepted. 2025.

[95] Romain Hennequin et al. “Spleeter: a fast and efficient music source separation tool with pre-trained models”. In: *J. Open Source Softw.* 5.56 (2020), p. 2154.

[96] Shawn Hershey et al. “CNN architectures for large-scale audio classification”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. IEEE, 2017, pp. 131–135.

[97] Aaron Hertzmann et al. "Image analogies". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001*. ACM, 2001, pp. 327–340.

[98] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[99] Dwayne Richard Hipp. *SQLite*. 2004. URL: https://www.sqlite.org.

[100] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[101] Ziming Hong et al. "Improving Non-Transferable Representation Learning by Harnessing Content and Style". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[102] Henry Hsu and Peter A Lachenbruch. "Paired t test". In: *Wiley StatsRef: statistics reference online* (2014).

[103] Wei-Ning Hsu, Yu Zhang, and James R. Glass. "Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, pp. 1878–1889.

[104] Edward J. Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[105] Cheng-Zhi Anna Huang et al. "Counterpoint by Convolution". In: *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*. 2017, pp. 211–218.

[106] Cheng-Zhi Anna Huang et al. “Music Transformer: Generating Music with Long-Term Structure”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[107] Sicong Huang et al. “TimbreTron: A WaveNet(CycleGAN(CQT(Audio))) Pipeline for Musical Timbre Transfer”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[108] Yu-Siang Huang and Yi-Hsuan Yang. “Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions”. In: *MM ’20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*. ACM, 2020, pp. 1180–1188.

[109] Yujia Huang et al. “Symbolic Music Generation with Non-Differentiable Rule Guided Diffusion”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[110] Yun-Ning Hung et al. “Musical Composition Style Transfer via Disentangled Timbre Representations”. In: *IJCAI*. 2019, pp. 4697–4703.

[111] Phillip Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 5967–5976.

[112] Dasaem Jeong et al. “Graph Neural Network for Music Score Data and Modeling Expressive Piano Performance”. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 3060–3070.

[113] Xu Ji, Andrea Vedaldi, and João F. Henriques. “Invariant Information Clustering for Unsupervised Image Classification and Segmentation”. In: *2019 IEEE/CVF International Con-*

ference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. IEEE, 2019, pp. 9864–9873.

[114] Junyan Jiang et al. “Large-vocabulary Chord Transcription Via Chord Structure Decomposition”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*. 2019, pp. 644–651.

[115] Jason Joven. *Minty Fresh: Spotify Makes a New Home for EDM*. 2018. URL: https://hmc.chartmetric.com/minty-fresh-spotify-makes-a-new-home-for-edm/.

[116] Hirokazu Kameoka et al. “StarGAN-VC: non-parallel many-to-many Voice Conversion Using Star Generative Adversarial Networks”. In: *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*. IEEE, 2018, pp. 266–273.

[117] Takuhiro Kaneko et al. “StarGAN-VC2: Rethinking Conditional Methods for StarGAN-Based Voice Conversion”. In: *20th Annual Conference of the International Speech Communication Association, Interspeech 2019, Graz, Austria, September 15-19, 2019*. ISCA, 2019, pp. 679–683.

[118] Tornike Karchkhadze, Mohammad Rasool Izadi, and Shlomo Dubnov. “Simultaneous Music Separation and Generation Using Multi-Track Latent Diffusion Models”. In: *2025 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2025, Hyderabad, India, April 6-11, 2025*. IEEE, 2025, pp. 1–5.

[119] Tornike Karchkhadze et al. “Multi-Track MusicLDM: Towards Versatile Music Generation with Latent Diffusion Model”. In: *CoRR* abs/2409.02845 (2024).

[120] Tero Karras, Samuli Laine, and Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 4401–4410.

[121] Kevin Kilgour et al. “Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms”. In: *20th Annual Conference of the International Speech Communication Association, Interspeech 2019, Graz, Austria, September 15-19, 2019*. ISCA, 2019, pp. 2350–2354.

[122] Hyunjik Kim and Andriy Mnih. “Disentangling by Factorising”. In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2654–2663.

[123] Taeksoo Kim et al. “Learning to Discover Cross-Domain Relations with Generative Adversarial Networks”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1857–1865.

[124] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[125] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014.

[126] Phillip B. Kirlin and Paul E. Utgoff. “A Framework for Automated Schenkerian Analysis”. In: *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*. 2008, pp. 363–368.

[127] Heinrich Christoph Koch. *Versuch einer Anleitung zur Composition*. Vol. 72. bey Adam Friedrich Böhme, 1787.

[128] Eunjeong Stella Koh, Shlomo Dubnov, and Dustin Wright. “Rethinking Recurrent Latent Variable Model for Music Composition”. In: *20th IEEE International Workshop on Multimedia Signal Processing, MMSP 2018, Vancouver, BC, Canada, August 29-31, 2018*. IEEE, 2018, pp. 1–6.

[129] Qiuqiang Kong et al. “High-Resolution Piano Transcription With Pedals by Regressing Onset and Offset Times”. In: *IEEE ACM Trans. Audio Speech Lang. Process.* 29 (2021), pp. 3707–3717.

[130] Zhifeng Kong et al. “Audio Flamingo: A Novel Audio Language Model with Few-Shot Learning and Dialogue Abilities”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[131] Zhifeng Kong et al. “Audio Flamingo: A Novel Audio Language Model with Few-Shot Learning and Dialogue Abilities”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[132] Florian Krebs, Sebastian Böck, and Gerhard Widmer. “Rhythmic Pattern Modeling for Beat and Downbeat Tracking in Musical Audio”. In: *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*. 2013, pp. 227–232.

[133] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Commun. ACM* 60.6 (2017), pp. 84–90.

[134] Carol L. Krumhansl. *Cognitive Foundations of Musical Pitch*. Oxford University Press, Nov. 2001. ISBN: 9780195148367.

[135] Gihyun Kwon and Jong Chul Ye. “CLIPstyler: Image Style Transfer with a Single Text Condition”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 2022, pp. 18041–18050.

[136] Ohsung Kwon et al. “Emotional Speech Synthesis Based on Style Embedded Tacotron2 Framework”. In: *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*. 2019, pp. 1–4.

[137] Kushal Lakhotia et al. “On generative spoken language modeling from raw audio”. In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 1336–1354.

[138] Stefan Lattner, Monika Dörfler, and Andreas Arzt. “Learning Complex Basis Functions for Invariant Representations of Audio”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*. 2019, pp. 700–707.

[139] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. “Imposing higher-level Structure in Polyphonic Music Generation using Convolutional Restricted Boltzmann Machines and Constraints”. In: *CoRR* abs/1612.04742 (2016).

[140] Yann LeCun. “A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27”. In: *Open Review* 62.1 (2022).

[141] Fred Lerdahl and Ray S Jackendoff. *A Generative Theory of Tonal Music*. MIT press, 1996.

[142] Junnan Li et al. “Align before Fuse: Vision and Language Representation Learning with Momentum Distillation”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*. 2021, pp. 9694–9705.

[143] Junnan Li et al. “BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models”. In: *International Conference on Machine Learning, ICML 2023*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 19730–19742.

[144] Kenneth Li et al. "Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023.

[145] Shuyu Li and Yunsick Sung. "MelodyDiffusion: Chord-Conditioned Melody Generation Using a Transformer-Based Diffusion Model". In: *Mathematics* 11.8 (2023), p. 1915.

[146] Yinghao Aaron Li, Ali Zare, and Nima Mesgarani. "StarGANv2-VC: A Diverse, Unsupervised, Non-Parallel Framework for Natural-Sounding Voice Conversion". In: *22nd Annual Conference of the International Speech Communication Association, Interspeech 2021, Brno, Czechia, August 30 - September 3, 2021.* ISCA, 2021, pp. 1349–1353.

[147] Yingzhen Li and Stephan Mandt. "Disentangled Sequential Autoencoder". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018.* Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 5656–5665.

[148] Yizhi Li et al. "MERT: Acoustic Music Understanding Model with Large-Scale Self-supervised Training". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net, 2024.

[149] *Liberated Pixel Cup.* http://lpc.opengameart.org/. [Accessed 26-09-2024].

[150] Liwei Lin et al. "A unified model for zero-shot music source separation, transcription and synthesis". In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021.* 2021, pp. 381–388.

[151] Liwei Lin et al. "Arrange, Inpaint, and Refine: Steerable Long-term Music Audio Generation and Editing via Content-based Controls". In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024.* ijcai.org, 2024, pp. 7690–7698.

[152] Liwei Lin et al. "Content-Based Controls for Music Large Language Modeling". In: *Proceedings of the 25th International Society for Music Information Retrieval Conference, ISMIR 2024, San Francisco, California, USA and Online, November 10-14, 2024*. 2024, pp. 783–790.

[153] Ming-Yu Liu, Thomas M. Breuel, and Jan Kautz. "Unsupervised Image-to-Image Translation Networks". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, pp. 700–708.

[154] Shansong Liu et al. "Music Understanding LLaMA: Advancing Text-to-Music Generation with Question Answering and Captioning". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024, Seoul, Republic of Korea, April 14-19, 2024*. IEEE, 2024, pp. 286–290.

[155] Xuanjie Liu et al. "Learning Interpretable Low-dimensional Representation via Physical Symmetry". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. 2023.

[156] Yang Liu et al. "Multi-Task Adversarial Network for Disentangled Feature Learning". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 3743–3751.

[157] Francesco Locatello et al. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 4114–4124.

[158] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[159] Chien-Yu Lu et al. “Play as You Like: Timbre-Enhanced Multi-Modal Music Style Transfer”. In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 1061–1068.

[160] Peiling Lu et al. “MuseCoco: Generating Symbolic Music from Text”. In: *CoRR* abs/2306.00110 (2023).

[161] Andreas Lugmayr et al. “RePaint: Inpainting using Denoising Diffusion Probabilistic Models”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 2022, pp. 11451–11461.

[162] Yin-Jyun Luo, Kat Agres, and Dorien Herremans. “Learning Disentangled Representations of Timbre and Pitch for Musical Instrument Sounds Using Gaussian Mixture Variational Autoencoders”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*. 2019, pp. 746–753.

[163] Yin-Jyun Luo, Sebastian Ewert, and Simon Dixon. “Towards Robust Unsupervised Disentanglement of Sequential Data - A Case Study Using Music Audio”. In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*. ijcai.org, 2022, pp. 3299–3305.

[164] Yin-Jyun Luo, Sebastian Ewert, and Simon Dixon. “Unsupervised Pitch-Timbre Disentanglement of Musical Instruments Using a Jacobian Disentangled Sequential Autoencoder”.

In: *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2024, pp. 1036–1040.

[165] Wenye Ma, Xinyue Li, and Gus Xia. "Do music LLMs learn symbolic concepts? A pilot study using probing and intervention". In: *Audio Imagination: NeurIPS 2024 Workshop AI-Driven Speech, Music, and Sound Generation*. 2024.

[166] Wenye Ma and Gus Xia. "Exploring the Internal Mechanisms of Music LLMs: A Study of Root and Quality via Probing and Intervention Techniques". In: *ICML 2024 Workshop on Mechanistic Interpretability*. 2024.

[167] Martin E. Malandro. "Composer's Assistant 2: Interactive Multi-Track MIDI Infilling With Fine-Grained User Control". In: *Proceedings of the 25th International Society for Music Information Retrieval Conference, ISMIR 2024, San Francisco, California, USA and Online, November 10-14, 2024*. 2024, pp. 438–445.

[168] Martin E. Malandro. "Composer's Assistant: An Interactive Transformer for Multi-Track MIDI Infilling". In: *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023, Milan, Italy, November 5-9, 2023*. 2023, pp. 327–334.

[169] Huanru Henry Mao, Taylor Shin, and Garrison W. Cottrell. "DeepJ: Style-Specific Music Generation". In: *12th IEEE International Conference on Semantic Computing, ICSC 2018, Laguna Hills, CA, USA, January 31 - February 2, 2018*. IEEE Computer Society, 2018, pp. 377–382.

[170] Alan Marsden. "Schenkerian analysis by computer: A proof of concept". In: *Journal of New Music Research* 39.3 (2010), pp. 269–289.

[171] Michaël Mathieu et al. "Disentangling factors of variation in deep representation using adversarial training". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 5041–5049.

[172] Michael McAuliffe et al. "Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi". In: *18th Annual Conference of the International Speech Communication Association, Interspeech 2017, Stockholm, Sweden, August 20-24, 2017.* ISCA, 2017, pp. 498–502.

[173] Jan Melechovský et al. "Mustango: Toward Controllable Text-to-Music Generation". In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024.* Association for Computational Linguistics, 2024, pp. 8293–8316.

[174] Lejun Min et al. "Polyffusion: A Diffusion Model for Polyphonic Score Generation With Internal and External Controls". In: *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023, Milan, Italy, November 5-9, 2023.* 2023, pp. 231–238.

[175] Gautam Mittal et al. "Symbolic Music Generation with Diffusion Models". In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021.* 2021, pp. 468–475.

[176] Olof Mogren. "C-RNN-GAN: Continuous recurrent neural networks with adversarial training". In: *CoRR* abs/1611.09904 (2016).

[177] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel.* Omnipress, 2010, pp. 807–814.

[178] Eita Nakamura and Kazuyoshi Yoshii. "Statistical piano reduction controlling performance difficulty". In: *APSIPA Transactions on Signal and Information Processing* 7 (2018).

[179] Eugene Narmour. *The analysis and cognition of basic melodic structures: The implication-realization model.* University of Chicago Press, 1990.

[180] Daiki Naruse et al. “Pop Music Generation with Controllable Phrase Lengths”. In: *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022, Bengaluru, India, December 4-8, 2022.* 2022, pp. 125–131.

[181] Meike Nauta et al. “From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI”. In: *ACM Comput. Surv.* 55.13s (2023), 295:1–295:42.

[182] Yuval Netzer et al. “Reading digits in natural images with unsupervised feature learning”. In: *NIPS workshop on deep learning and unsupervised feature learning.* Vol. 2011. 2. Granada. 2011, p. 4.

[183] Stephen Ni-Hahn et al. “A New Dataset, Notation Software, and Representation for Computational Schenkerian Analysis”. In: *Proceedings of the 25th International Society for Music Information Retrieval Conference, ISMIR 2024, San Francisco, California, USA and Online, November 10-14, 2024.* 2024, pp. 866–873.

[184] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. “Neural Discrete Representation Learning”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA.* 2017, pp. 6306–6315.

[185] OpenAI. “GPT-4 Technical Report”. In: *CoRR* abs/2303.08774 (2023).

[186] Nicola Orio and Antonio Rodà. “A Measure of Melodic Similarity based on a Graph Representation of the Music Structure”. In: *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009.* International Society for Music Information Retrieval, 2009, pp. 543–548.

[187] Longshen Ou et al. “Exploring Transformer’s Potential on Automatic Piano Transcription”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022*. IEEE, 2022, pp. 776–780.

[188] François Pachet, Alexandre Papadopoulos, and Pierre Roy. “Sampling Variations of Sequences for Structured Music Generation”. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*. 2017, pp. 167–173.

[189] Vassil Panayotov et al. “Librispeech: An ASR corpus based on public domain audio books”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*. IEEE, 2015, pp. 5206–5210.

[190] Bryan Pardo and William P. Birmingham. “Algorithms for Chordal Analysis”. In: *Computer Music Journal* 26.2 (2002), pp. 27–49.

[191] Taesung Park et al. “Contrastive Learning for Unpaired Image-to-Image Translation”. In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part IX*. Vol. 12354. Lecture Notes in Computer Science. Springer, 2020, pp. 319–345.

[192] Or Patashnik et al. “StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery”. In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 2065–2074.

[193] Ashis Pati, Alexander Lerch, and Gaëtan Hadjeres. “Learning to Traverse Latent Spaces for Musical Score Inpainting”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*. 2019, pp. 343–351.

[194] Judea Pearl. *Causality*. Cambridge university press, 2009.

[195] Graham Percival, Satoru Fukayama, and Masataka Goto. "Song2Quartet: A System for Generating String Quartet Cover Songs from Polyphonic Audio of Popular Music". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015.* 2015, pp. 114–120.

[196] Ben Poole et al. "On variational bounds of mutual information". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5171–5180.

[197] Kaizhi Qian et al. "AutoVC: Zero-Shot Voice Style Transfer with Only Autoencoder Loss". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 5210–5219.

[198] Kaizhi Qian et al. "F0-Consistent Many-To-Many Non-Parallel Voice Conversion Via Conditional Autoencoder". In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020.* IEEE, 2020, pp. 6284–6288.

[199] Kaizhi Qian et al. "Unsupervised Speech Decomposition via Triple Information Bottleneck". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event.* Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 7836–7846.

[200] Xingwei Qu et al. "MuPT: A Generative Symbolic Music Pretrained Transformer". In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025.* OpenReview.net, 2025.

[201] Colin Raffel et al. "MIR_EVAL: A Transparent Implementation of Common MIR Metrics". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014.* 2014, pp. 367–372.

[202] Yi Ren et al. “PopMAG: Pop Music Accompaniment Generation”. In: *MM ’20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*. ACM, 2020, pp. 1198–1206.

[203] Adam 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, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4361–4370.

[204] John Rothgeb. *Introduction to the theory of Heinrich Schenker: the nature of the musical work of art*. New York: Longman, 1982.

[205] Dimitri von Rütte et al. “FIGARO: Controllable Music Generation using Learned and Expert Features”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[206] Patsorn Sangkloy et al. “Scribbler: Controlling Deep Image Synthesis with Sketch and Color”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 6836–6845.

[207] Franco Scarselli et al. “The Graph Neural Network Model”. In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80.

[208] Henry Scheffe. *The analysis of variance*. Vol. 72. John Wiley & Sons, 1999.

[209] Heinrich Schenker. *Free Composition (Der freie Satz)*. Trans. by Ernst Oster. Translated and edited by Ernst Oster. New York: Longman, 1979.

[210] Flavio Schneider, Zhijing Jin, and Bernhard Schölkopf. “Moûsai: Text-to-Music Generation with Long-Context Latent Diffusion”. In: *CoRR* abs/2301.11757 (2023).

[211] Arnold Schoenberg. *Theory of harmony*. Univ of California Press, 1983.

[212] Bernhard Schölkopf et al. “Toward Causal Representation Learning”. In: *Proc. IEEE* 109.5 (2021), pp. 612–634.

[213] Ian Simon, Dan Morris, and Sumit Basu. “MySong: automatic accompaniment generation for vocal melodies”. In: *Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008*. ACM, 2008, pp. 725–734.

[214] Ian Simon and Sageev Oore. *Performance RNN: Generating Music with Expressive Timing and Dynamics*. https://magenta.tensorflow.org/performance-rnn. Blog. 2017.

[215] Ian Simon et al. “Learning a Latent Space of Multitrack Measures”. In: *CoRR* abs/1806.00195 (2018).

[216] Federico Simonetta et al. “Symbolic Music Similarity through a Graph-Based Representation”. In: *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion, Wrexham, United Kingdom, September 12-14, 2018*. ACM, 2018, 26:1–26:7.

[217] Stephen W. Smoliar. “A computer aid for Schenkerian analysis”. In: *Proceedings of the 1979 Annual Conference, Detroit, Michigan, USA, October 29-31, 1979*. ACM, 1979, pp. 110–115.

[218] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958.

[219] Philip Tagg. “Analysing popular music: theory, method and practice”. In: *Popular music* 2 (1982), pp. 37–67.

[220] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. “Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 2015, pp. 1556–1566.

[221] Yaniv Taigman, Adam Polyak, and Lior Wolf. "Unsupervised Cross-Domain Image Generation". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[222] Hirofumi Takamori et al. "Audio-Based Automatic Generation of a Piano Reduction Score by Considering the Musical Structure". In: *MultiMedia Modeling - 25th International Conference, MMM 2019, Thessaloniki, Greece, January 8-11, 2019, Proceedings, Part II*. Vol. 11296. Lecture Notes in Computer Science. Springer, 2019, pp. 169–181.

[223] Hirofumi Takamori et al. "Automatic arranging musical score for piano using important musical elements". In: *Proceedings of the Sound and Music Computing Conference (SMC)*. 2017, pp. 35–41.

[224] Chih-Pin Tan, Shuen-Huei Guan, and Yi-Hsuan Yang. "PiCoGen: Generate Piano Covers with a Two-stage Approach". In: *Proceedings of the 2024 International Conference on Multimedia Retrieval, ICMR 2024*. ACM, 2024, pp. 1180–1184.

[225] Chih-Pin Tan et al. "PiCoGen2: Piano Cover Generation With Transfer Learning Approach and Weakly Aligned Data". In: *Proceedings of the 25th International Society for Music Information Retrieval Conference, ISMIR 2024*. 2024, pp. 555–562.

[226] Hao Tan et al. "VIMPAC: Video Pre-Training via Masked Token Prediction and Contrastive Learning". In: *CoRR* abs/2106.11250 (2021).

[227] Hao Hao Tan. *Frechet Audio Distance in PyTorch*. https://github.com/gudgud96/frechet-audio-distance. 2022.

[228] Adly Templeton et al. "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet". In: *Transformer Circuits Thread* (2024).

[229] John Thickstun, Zaïd Harchaoui, and Sham M. Kakade. "Learning Features of Music From Scratch". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[230] John Thickstun et al. “Anticipatory Music Transformer”. In: *Transactions on Machine Learning Research* 2024 (2024).

[231] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. “Transformer VQ-VAE for Unsupervised Unit Discovery and Speech Synthesis: ZeroSpeech 2020 Challenge”. In: *21st Annual Conference of the International Speech Communication Association, Interspeech 2020, Virtual Event, Shanghai, China, October 25-29, 2020.* ISCA, 2020, pp. 4851–4855.

[232] Andros Tjandra et al. “Unsupervised Learning of Disentangled Speech Content and Style Representation”. In: *22nd Annual Conference of the International Speech Communication Association, Interspeech 2021, Brno, Czechia, August 30 - September 3, 2021.* ISCA, 2021, pp. 4089–4093.

[233] Satoshi Tojo, Keiji Hirata, and Masatoshi Hamanaka. “Computational Reconstruction of Cognitive Music Theory”. In: *New Gener. Comput.* 31.2 (2013), pp. 89–113.

[234] Nikzad Benny Toomarian and Jacob Barhen. “Learning a Trajectory Using Adjoint Functions and Teacher Forcing”. In: *Neural Networks* 5.3 (1992), pp. 473–484.

[235] Hugo Touvron et al. “LLaMA: Open and Efficient Foundation Language Models”. In: *CoRR* abs/2302.13971 (2023).

[236] Christopher J. Tralie. “Cover Song Synthesis by Analogy”. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018.* 2018, pp. 197–203.

[237] Sandra E Trehub and Erin E Hannon. “Infant Music Perception: Domain-General or Domain-Specific Mechanisms?” In: *Cognition* 100.1 (2006), pp. 73–99.

[238] George Tzanetakis and Perry R. Cook. “Musical genre classification of audio signals”. In: *IEEE Trans. Speech Audio Process.* 10.5 (2002), pp. 293–302.

[239] Mohammad Hassan Vali and Tom Bäckström. "Interpretable Latent Space Using Space-Filling Curves for Phonetic Analysis in Voice Conversion". In: *24th Annual Conference of the International Speech Communication Association, Interspeech 2023, Dublin, Ireland, August 20-24, 2023*. ISCA, 2023, pp. 306–310.

[240] Marcel A. Vélez Vásquez et al. "Exploring the Inner Mechanisms of Large Generative Music Models". In: *Proceedings of the 25th International Society for Music Information Retrieval Conference, ISMIR 2024*. 2024, pp. 791–798.

[241] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, pp. 5998–6008.

[242] Bryan Wang and Yi-Hsuan Yang. "PerformanceNet: Score-to-Audio Music Generation with Multi-Band Convolutional Residual Network". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 1174–1181.

[243] Ziyu Wang, Lejun Min, and Gus Xia. "Whole-Song Hierarchical Generation of Symbolic Music Using Cascaded Diffusion Models". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[244] Ziyu Wang and Gus Xia. "MuseBERT: Pre-training Music Representation for Music Understanding and Controllable Generation". In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*. 2021, pp. 722–729.

[245] Ziyu Wang et al. "Audio-To-Symbolic Arrangement Via Cross-Modal Music Representation Learning". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022.* IEEE, 2022, pp. 181–185.

[246] Ziyu Wang et al. "Automatic Melody Reduction via Shortest Path Finding". In: *Proceedings of the 26th International Society for Music Information Retrieval Conference.* Accepted. 2025.

[247] Ziyu Wang et al. "Learning Interpretable Representation for Controllable Polyphonic Music Generation". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference.* 2020, pp. 662–669.

[248] Ziyu Wang et al. "PIANOTREE VAE: Structured Representation Learning for Polyphonic Music". In: *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020, Montreal, Canada, October 11-16, 2020.* 2020, pp. 368–375.

[249] Ziyu Wang et al. "POP909: A Pop-Song Dataset for Music Arrangement Generation". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference.* 2020, pp. 38–45.

[250] Ziyu Wang et al. "POP909: A Pop-song Dataset for Music Arrangement Generation". In: *Proceedings of 21st International Conference on Music Information Retrieval (ISMIR), virtual conference.* 2020.

[251] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques.* Addison-Wesley, ACM Press, 1992.

[252] Megan Wei et al. "Do Music Generation Models Encode Music Theory?" In: *Proceedings of the 25th International Society for Music Information Retrieval Conference, ISMIR 2024.* 2024, pp. 680–687.

[253] Shiqi Wei and Gus Xia. "Learning long-term music representations via hierarchical contextual constraints". In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021.* 2021, pp. 738–745.

[254] Shiqi Wei et al. “Music Phrase Inpainting Using Long-Term Representation and Contrastive Loss”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*. IEEE, 2022, pp. 186–190.

[255] Peter West et al. “The Generative AI Paradox: "What It Can Create, It May Not Understand"”. In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[256] Gerhard Widmer, Sebastian Flossmann, and Maarten Grachten. “YQX plays Chopin”. In: *AI magazine* 30.3 (2009), pp. 35–35.

[257] Gerhard Widmer and Asmir Tobudic. “Playing Mozart by analogy: Learning multi-level timing and dynamics strategies”. In: *Journal of New Music Research* 32.3 (2003), pp. 259–268.

[258] Richard Wilhelm, Cary F Baynes, and Carl G Jung. *I Ching: Book of changes*. Grange Books, 2001.

[259] Shih-Lun Wu et al. “Music ControlNet: Multiple Time-Varying Controls for Music Generation”. In: *IEEE ACM Trans. Audio Speech Lang. Process.* 32 (2024), pp. 2692–2703.

[260] Yiming Wu et al. “Semi-supervised Neural Chord Estimation Based on a Variational Autoencoder with Discrete Labels and Continuous Textures of Chords”. In: *CoRR* abs/2005.07091 (2020).

[261] Yusong Wu et al. “Large-Scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*. IEEE, 2023, pp. 1–5.

[262] Yuxuan Wu et al. “Unsupervised Disentanglement of Content and Style via Variance-Invariance Constraints”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[263] Shaoan Xie, Qirong Ho, and Kun Zhang. “Unsupervised Image-to-Image Translation with Density Changing Regularization”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. 2022.

[264] Shaoan Xie et al. “Multi-domain image generation and translation with identifiability guarantees”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[265] Yangchen Xie et al. “DG-Font: Deformable Generative Networks for Unsupervised Font Generation”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 5130–5140.

[266] Eric Xing et al. “Critiques of World Models”. In: *CoRR* abs/2507.05169 (2025).

[267] Ruibin Xiong et al. “On Layer Normalization in the Transformer Architecture”. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 10524–10533.

[268] Wilson Yan et al. “VideoGPT: Video Generation using VQ-VAE and Transformers”. In: *CoRR* abs/2104.10157 (2021).

[269] Xinchen Yan et al. “Attribute2image: Conditional image generation from visual attributes”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 776–791.

[270] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. “MidiNet: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation”. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*. 2017, pp. 324–331.

[271] Ruihan Yang et al. “Deep Music Analogy Via Latent Representation Disentanglement”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*. 2019, pp. 596–603.

[272] Ruihan Yang et al. “Inspecting and Interacting with Meaningful Music Representations using VAE”. In: *19th International Conference on New Interfaces for Musical Expression, NIME 2019, Porto Alegre, Brazil, June 3-6, 2019*. nime.org, 2019, pp. 307–312.

[273] Jin Y Yen. “Finding the k shortest loopless paths in a network”. In: *Management Science* 17.11 (1971), pp. 712–716.

[274] Lantao Yu et al. “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient”. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. AAAI Press, 2017, pp. 2852–2858.

[275] Ruibin Yuan et al. “ChatMusician: Understanding and Generating Music Intrinsically with LLM”. In: *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*. Association for Computational Linguistics, 2024, pp. 6252–6271.

[276] Ruibin Yuan et al. “YuE: Scaling Open Foundation Models for Long-Form Music Generation”. In: *CoRR* abs/2503.08638 (2025).

[277] Yongyi Zang and Yixiao Zhang. “The Interpretation Gap in Text-to-Music Generation Models”. In: *Proceedings of the 3rd Workshop on NLP for Music and Audio (NLP4MusA)*. Oakland, USA: Association for Computational Lingustics, Nov. 2024, pp. 112–118.

[278] Mingliang Zeng et al. “MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training”. In: *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021*. Vol. ACL/IJCNLP 2021. Findings of ACL. Association for Computational Linguistics, 2021, pp. 791–800.

[279] Wei Zeng, Xian He, and Ye Wang. "End-to-End Real-World Polyphonic Piano Audio-to-Score Transcription with Hierarchical Decoding". In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024*. ijcai.org, 2024, pp. 7788–7795.

[280] Huan Zhang and Simon Dixon. "Disentangling the Horowitz Factor: Learning Content and Style From Expressive Piano Performance". In: *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*. IEEE, 2023, pp. 1–5.

[281] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. "Adding Conditional Control to Text-to-Image Diffusion Models". In: *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 2023, pp. 3813–3824.

[282] Yixiao Zhang et al. "BUTTER: A Representation Learning Framework for Bi-directional Music-Sentence Retrieval and Generation". In: *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*. Online: Association for Computational Linguistics, Oct. 2020, pp. 54–58.

[283] Yixiao Zhang et al. "Instruct-MusicGen: Unlocking Text-to-Music Editing for Music Language Models via Instruction Tuning". In: *CoRR* abs/2405.18386 (2024).

[284] Yixiao Zhang et al. "MusicMagus: Zero-Shot Text-to-Music Editing via Diffusion Models". In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*. ijcai.org, 2024, pp. 7805–7813.

[285] Yu Zhang et al. "A Survey on Neural Network Interpretability". In: *IEEE Trans. Emerg. Top. Comput. Intell.* 5.5 (2021), pp. 726–742.

[286] Jingwei Zhao, Gus Xia, and Ye Wang. "Q&A: Query-Based Representation Learning for Multi-Track Symbolic Music Re-Arrangement". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 2023, pp. 5878–5886.

[287] Jingwei Zhao et al. “Structured Multi-Track Accompaniment Arrangement via Style Prior Modelling”. In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. 2024.

[288] Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. “Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Association for Computational Linguistics, 2017, pp. 654–664.

[289] Hongyuan Zhu et al. “XiaoIce Band: A Melody and Arrangement Generation Framework for Pop Music”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. ACM, 2018, pp. 2837–2846.

[290] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2242–2251.

[291] Yi Zou et al. “Melons: Generating Melody With Long-Term Structure Using Transformers And Structure Graph”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*. IEEE, 2022, pp. 191–195.