A Neural Score Follower for Computer Accompaniment of Polyphonic Musical Instruments

by

Ashwin Pillay

Submitted to the School of Music
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MUSIC AND TECHNOLOGY

at

CARNEGIE MELLON UNIVERSITY

May 2024

Authored by: Ashwin Pillay
School of Music
May 16, 2024

Certified by: Dr. Richard M. Stern
Professor of Electrical and Computer Engineering, Thesis Supervisor

Certified by: Dr. Roger B. Dannenberg
Emeritus Professor of Computer Science, Art and Music, Thesis Supervisor

A Neural Score Follower for Computer Accompaniment of Polyphonic Musical Instruments

by

Ashwin Pillay

Submitted to the School of Music
on May 16, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MUSIC AND TECHNOLOGY

ABSTRACT

Real-time computer-based accompaniment for human musical performances entails three critical tasks: identifying what the performer is playing, locating their position within the score, and synchronously playing the accompanying parts. Among these, the second task (score following) has been addressed through methods such as dynamic programming on string sequences, Hidden Markov Models (HMMs), and Online Time Warping (OLTW). Yet, the remarkably successful techniques of Deep Learning (DL) have not been directly applied to this problem.

Therefore, we introduce **HeurMiT**, a novel DL-based score-following framework, utilizing a neural architecture designed to learn compressed latent representations that enables precise performer tracking despite deviations from the score. Parallelly, we implement a real-time MIDI data augmentation toolkit, aimed at enhancing the robustness of these learned representations. Additionally, we integrate the overall system with simple heuristic rules to create a comprehensive framework that can interface seamlessly with existing transcription and accompaniment technologies.

However, thorough experimentation reveals that despite its impressive computational efficiency, HeurMiT's underlying limitations prevent it from being practical in real-world score following scenarios. Consequently, we present our work as an introductory exploration into the world of DL-based score followers, while highlighting some promising avenues to encourage future research towards robust, state-of-the-art neural score following systems.

Thesis supervisor: Dr. Richard M. Stern
Title: Professor of Electrical and Computer Engineering

Thesis supervisor: Dr. Roger B. Dannenberg
Title: Emeritus Professor of Computer Science, Art and Music

Contents

List of Figures

List of Tables

Acronyms

AE Alignment Error.

AMT Automatic Music Transcription.

API Application Programming Interface.

CLI command-line interface.

CNN Convolutional Neural Network.

CQT Constant-Q Transform.

CRF Conditional Random Fields.

DAW Digital Audio Workstation.

DL Deep Learning.

DNN Deep Neural Networks.

DRL Deep Reinforcement Learning.

DSP Digital Signal Processing.

DTW Dynamic Time Warping.

GDL Geometric Deep Learning.

GIL Global Interpreter Lock.

HMM Hidden Markov Models.

LSTM Long Short-Term Memory.

MDTK MIDI Degradation Toolkit.

MIREX Music Information Retrieval Evaluation eXchange.

NMF Non-negative Matrix Factorization.

NSGT Non Stationary Gabor Transform.

OLTW Online Time Warping.

OSC Open Sound Control.

ReLU Rectified Linear Unit.

RL Reinforcement Learning.

SD standard deviation.

STFT Short-time Fourier Transform.

UDP User Datagram Protocol.

Chapter 1

Introduction

Music has always been a dynamic art form that readily embraces technological innovations to expand its expressivity and accessibility. The advent of modern computing has transformed music production from a resource-intensive process, which traditionally required significant manpower, expertise, and financial investment, into an activity that individuals can pursue effectively within the comfort of their own rooms.

However, computers are yet to be widely applied in live collaborations with human musicians. Enabling computers to act as accompanists to human performers introduces a new dimension to live performances. They could simultaneously control multiple instruments or even non-musical elements such as lighting and stage visuals. Crucially, we strongly intend these collaborative performances to be human-led; *i.e.*, while both the human performer and the computer have access to the score, the human dictates the timing and tempo of the performance. The computer must listen to the human's performance and play the rest of the score, intelligently navigating through any mistakes or improvisations made by the performer. This setup allows human musicians to maintain their creative expression while benefiting from the quick, error-free response, and multitasking capabilities of computer systems.

The problem of computer-based accompaniment has been the subject of previous research. In 1984, Dannenberg [1] identified the main challenges in this domain, segmenting them into three sub-problems: identifying what a performer is playing, locating their position within the score (termed *score following*), and playing the remaining score components to accompany the performer with minimal latency. With respect to score following, various strategies ranging from dynamic programming to Hidden Markov Models (HMM), have been employed but the potential of Deep Learning (DL) remains to be fully leveraged. Recent advances in DL have revolutionized fields such as text understanding [2], speech recognition [3], and image generation [4], suggesting promising new directions for enhancing score following systems.

In this work, we develop **HeurMiT**, a score following system that comprises the following:

1. **Tyke**, a compact Deep Neural Networks (DNN)-based architecture trained using a template-matching DL paradigm. Tyke aims to learn compressed latent feature representations of the performance and the score while being robust against any performer-induced deviations.

2. A set of common-sense heuristic rules that critique and safeguard Tyke's predictions, independently of any performance-specific nuances.

For training Tyke models to learn features robust to the performer's random deviations from the score, we have also developed an on-the-fly MIDI data augmentation library called **MIDIOgre**. This tool can synthetically generate a wide variety of commonly observed performance imperfections, which we incorporate into Tyke's training paradigm.

The chief benefits of HeurMiT include its $\mathcal{O}(1)$ performance. Additionally, its design enables easy integration with existing solutions addressing the other sub-problems outlined by Dannenberg. However, a thorough evaluation of its score following ability suggests that HeurMiT is **not ready** for real-world applications. In a best-case scenario, HeurMiT's prediction errors are only slightly better than our baseline system (Flippy), while it loses track of the performance more frequently on average. Moreover, when the performance tempo significantly differs from the score, HeurMiT fails to follow the performance comprehensively. Additionally, the implemented set of heuristics continues to focus on performance nuances, necessitating optimizations per performance for optimal results. In light of these limitations, we discuss several directions for future research that either seek to improve the current template-matching paradigm or identify alternative DL-based approaches that are inherently tempo-insensitive.

We view our work as an introductory exploration into the capabilities of DL-based neural score following systems. By documenting our approach and acknowledging both the strengths and limitations of our efforts, we aspire to encourage future research aimed at identifying much more robust and efficient methods for tracking performances across a diverse array of instruments, genres, and styles using DNNs. We believe continued efforts in this space will eventually enable effective real-time musical collaboration between humans and computers.

The subsequent sections of this work are structured as follows: Chapter 2 discusses existing literature related to score following, computer accompaniment, and recent advancements in DL. Chapter 3 introduces a formal definition of our DNN-based score following problem, detailing the neural architectures and training paradigms employed, along with auxiliary components addressing the first and third sub-problems. Chapter 4 outlines the metrics and experiments designed to evaluate our system comprehensively against existing works and set a benchmark for future comparisons. Chapter 5 provides a detailed analysis of the performances of our system, highlighting the pros and cons of our approach. Chapter 6 presents a set of future research directions for neural score following systems. Finally, chapter 7 concludes our research.

Chapter 2

Literature Review

2.1 Foundational Work based on Dynamic Programming

The concept of *score following* and its extension to real-time applications, termed *computer accompaniment*, was independently introduced by Dannenberg [1] and Vercoe [5] in 1984. This pioneering work laid the foundation for pitch tracking-based score following systems, where incoming performance audio is analyzed through pitch detection—and, in certain cases, supported by additional data sources like fingering information and optical cues—to generate a sequence of musical notes.

Dannenberg's approach conceptualizes score following as a symbol-matching challenge, converting score and performance events into string sequences. The goal is to identify the least-cost alignment path between these sequences using dynamic programming techniques. Originally designed for monophonic instruments such as the trumpet, this method was later expanded to accommodate polyphonic performances [6], enhance resilience to performance interruptions through multiple matchers running in parallel [7], and track complex musical ornaments, including trills and glissandi [7]. Combining ideas from Dannenberg [1] and Vercoe [5], Vercoe and Puckette proposed an alternative approach based on least-cost matching [8]. This methodology has since evolved to include considerations for fixed-length note segments [9], the ability to match new notes with previously overlooked ones [10], and the integration of pitch detection with note onset and tempo tracking [11].

2.2 Statistical Approaches

A fundamental objective for score following systems is to remain resilient against the variability and inaccuracies inherent in live human performances, especially regarding timing. Early systems employed heuristic rules informed by musical theory and general common sense. However, subsequent generations have adopted probabilistic statistical models to enhance accuracy and flexibility. For instance, in 1997, Grubb and Dannenberg [12] introduced a method that uses a sliding window to generate a continuous probability density function over possible score positions for vocal performance tracking. Pardo and Birmingham [13] refined the least-cost matching strategy with a probabilistic model that accounts for audio-to-symbol transcription errors and introduces penalties for omitting significant portions of

the score.

2.2.1 Hidden Markov Models (HMM)

Among the statistical models employed in score following, HMMs have become particularly prominent. Raphael's 1999 [14] proposal of an HMM-based score follower that operates directly on audio spectral features—bypassing the need for pitch-to-symbol conversion—marked a significant advancement. This model was further developed to account for performer errors [15], support polyphonic music [16], and handle impromptu skips and repeats in the performance [17]. Subsequently, several performance improvement extensions have also ensued [18]–[25], with works being as recent as from 2023 [26].

2.3 Online Time Warping (OLTW)-based Approaches

Following the development of OLTW [27], a variant of Dynamic Time Warping (DTW) characterized by linear space and time complexity, there was an advent of a new subclass of score followers. These facilitate incremental alignment of real-time performance audio with a synthesized version of the score using their Short-time Fourier Transform (STFT) analyses. In this direction, studies have expanded upon the work by Dixon and Widmer [28] to be run online, while aiming to minimize the discrepancies between offline and online DTW alignments [29], [30]. Notably, many systems benchmarked by the Music Information Retrieval Evaluation eXchange (MIREX) *Real-time Audio to Score Alignment (a.k.a Score Following)* task utilize OLTW in conjunction with advanced feature extraction methods, such as chromagrams [31] and Non-negative Matrix Factorization (NMF) bases [32]. A recent study by Lee [33] also demonstrates the effectiveness of combining OLTW with Constant-Q Transform (CQT).

2.4 Breakthroughs in Deep Learning (DL) and Intersections with Computer Accompaniment

In parallel with the progression of score following methodologies, DL [34] has risen to prominence through leveraging neural networks as universal function approximators [35]. This approach has become the go-to solution for a plethora of challenges across unimodal and multimodal applications in text, vision, video, and audio domains, among others [2], [36]–[38]. A pivotal aim of DL models is the extraction of optimal features. This is usually achieved by training neural networks on extensive sets of input-output examples to derive a condensed representation within a learned latent space. Such extracted features are now integral to cutting-edge solutions tackling complex problems such as vision-language understanding [39], speech processing [40], and code generation [41]. Data augmentation stands as a critical strategy for deriving efficient latent representations, wherein training data is enriched through artificial modifications of minor information contributors, thereby rendering models invariant to their random real-world variabilities, including noise. In the vision domain, data augmentation strategies range from simple techniques like cropping, rotations,

and color space transformations [42] to more complex methods such as CutMix [43]. For audio and speech, basic techniques include random pitch shifting and time stretching [44], with advanced methods like SpecAugment [45] also proving beneficial. In contrast, the exploration of effective augmentation techniques for symbolic music (e.g., MIDI, MusicXML, piano rolls) remains nascent despite their increasing utility [46], [47]. However, the MIDI Degradation Toolkit (MDTK) [48] initiates an exploration into effective MIDI augmentations like random note number shifts, velocity shifts, and timing adjustments.

Recent advancements in computer accompaniment have also leveraged Deep Neural Networks (DNN)s. Jiang et al. introduced RL-Duet [49], a Reinforcement Learning (RL) based generation agent that devises a policy for musical note generation conditioned on prior human and machine inputs. This showcases RL agents' capability in maintaining long-term tempo coherence through maximizing discounted rewards. Wang et al. presented SongDriver [50], a real-time music generator employing a Transformer [51] and a Conditional Random Fields (CRF) model for anticipative melody accompaniment with minimal latency. These studies primarily focus on *score-free* accompaniment, posing a likelihood for it to significantly diverge from the score. A conceivable improvement involves anchoring these models on partial score information via a Music ControlNet [52] though more direct score-based training approaches might offer optimized solutions. In 2023, Peter [53] proposed an RL agent, utilizing an attention-based neural network [54], trained via Offline Deep Reinforcement Learning (DRL) to align new performance data with the current score window and the most recent performance data. For real-time score following, this method integrates a tempo extractor and heuristic rules to bolster overall robustness.

2.5 Evaluating Existing Research

This section builds on our analysis of existing methodologies in score following and computer accompaniment by outlining the desired characteristics within an optimal score follower. We examine how these characteristics have been incorporated into existing research and identify how they can be enhanced. While we aim to integrate all the features outlined in this section within our work, we also intend for them to serve as general benchmarks for future developments in the field.

2.5.1 Accessibility

Performance variability, discounting intentional improvisations, is a function of the performer's expertise. Novices may not adhere closely to the score, posing challenges for score following systems originally conceived for professional performances [1], [5], [6]. A broader user base, beyond professional musicians, could vastly benefit from using score following systems as an advanced alternative to traditional metronomes. To cater to this demographic, future systems should accommodate varying levels of performance expertise, allowing for substantial deviations as long as the overall piece remains recognizable. Additionally, ease of installation and operation on widely available computing systems is crucial for widespread accessibility.

2.5.2 Flexibility

While initial score followers focused on monophonic instruments [1], [5], [8], [14], they were subsequently extended for polyphonic music by methods such as grouping notes based on their relative proximities [6]. Additionally, specialized systems have been developed for vocal tracking [12], and some require iterative rehearsals [8] or score-specific training [13], [14], [32]. Looking forward, an ideal score follower would inherently support polyphonic and vocal performances, minimize dependency on domain-specific heuristics, and efficiently handle a wide array of musical styles and complexities without the need for customized training.

2.5.3 Robustness[1]

Many symbol-matching and OLTW-based algorithms are designed to align performances with scores at the individual note level. However, when performances are uniformly transposed relative to the score, this may pose unique challenges. Several approaches have been developed to account for predictable musical ornaments [6], [7], but performers may introduce arbitrary new ornaments. Additionally, significant tempo variations have historically been challenging to accommodate [16]. An effective score-following method would need to maintain accuracy amidst these variations. In this context, a holistic template-matching approach that compares collections of performance events with groups of score events might offer a promising solution.

2.5.4 Performance Metrics

Throughout the development of score following and computer accompaniment systems, the metrics used to evaluate their performance have significantly evolved. Initially, limitations in computational capabilities may have restricted early studies [1], [5], [6], [14] to qualitative evaluations or minimal quantitative analysis. Subsequent research introduced more standardized measures, such as Alignment Error (AE) [28], assessing the temporal discrepancy between a performance event as identified by the score follower and its actual timing in the score. In 2007, Cont et al. [55] expanded the repertoire to include metrics like system latency and precision rate, establishing them as benchmarks within the MIREX evaluation framework. These metrics have since become a staple in the field [31], [32]. Nonetheless, Lee's [33] critique of the MIREX evaluation methods highlighted significant limitations, prompting a call for alternative approaches. Despite this, further critical evaluations of the efficacy and applicability of these metrics remain scarce.

[1] In the context of computer accompaniment systems, the term "robustness" has traditionally referred to the system's ability to gracefully recover from drastic jumps in score positions during a performance. In our discussion, however, we define robustness as the system's capacity to tolerate variations in the performance relative to the score.

Chapter 3

The Neural Score Following System

Addressing the challenges highlighted in existing score following research, as discussed in section 2.5, we formulate a new approach that leverages DL. Our strategy involves the conversion of score and performance data into piano roll formats, leveraging DNN models to extract robust latent features from them, and applying cross-correlation combined with practical heuristics to jointly compare recent performance data against likely score positions. Furthermore, we explore the DNN architecture integral to our framework, elaborating on the supervised learning paradigms utilized for its training. Additionally, we suggest methods to integrate our solution with existing components to enable the accompaniment of polyphonic instruments and vocal performances across a diverse array of musical genres.

3.1 Terminologies

Aligning with the terminology set forth by Bloch and Dannenberg [6], we define some key terms for describing our system: the input polyphonic instrument recording is referred to as the *solo*, and the system's output is termed the *accompaniment*. The piece being performed is called the *performance*, with its machine-readable form known as the *score*. Deviations by the performer from the score, which may include tempo variations or unintentional notes, are identified as *imperfections*.

3.2 Problem Definition

Externally, both the solo and accompaniment manifest as audio signals. However, our score-follower internally interprets piano-roll representations of the solo and the score. Piano rolls are favored for their intrinsic ability to depict notes in both a polyphonic and octave-sensitive manner, while accommodating various musical ornaments such as trills and nachschlags. The conversion from MIDI[1] and MusicXML to piano rolls is straightforward, with established methods also facilitating their derivation from waveform audios and STFTs. Mathematically, piano rolls can be represented by two-dimensional vectors, $p \in \mathbb{R}^{128 \times n}$, where 128

[1] https://midi.org/specs

corresponds to the full range of MIDI note numbers[2], and n represents the discrete temporal dimension. This structure allows for the efficient application of parallel vector operations available through computational tools like PyTorch [56].

Building upon the advantages of template-matching, as explored in section 2.5.3, we conceptualize score following as the challenge of locating the most fitting placement of the solo (template) within the larger score (target). Further, aligning with some existing works [1], [28], we hypothesize that the solo's most probable locations are within a vicinity of its last known position, hence optimizing efficiency by limiting the scope of template-matching to this specific area. This segment of the score, designated as the *context* ($C \in \mathbb{R}^{128\times c}$), alongside the latest collection of notes from the solo, known as the *window* ($W \in \mathbb{R}^{128\times w}$), forms the crux of our analytical framework. Here, c and w indicate the durations of score and solo analyzed analyzed for score following, respectively[3]. Our score-following DNN processes these vector representations as its primary inputs.

3.2.1 Learning Effective Latent Representations

When performances are perfect, Digital Signal Processing (DSP) techniques like cross-correlation are effective for template matching. However, our system must be capable of accurate prediction in the presence of the performer's imperfections. Within the piano-roll framework, common imperfections [48] such as note additions, deletions, delays, and pitch shifts appear as minor vector variations. Transpositions and localized tempo changes can also be represented by correlated pitch shifts and note delays, respectively. We intend the score follower to be insensitive to these variations, while focussing only on the information necessary to reliably locate the window within the score. Additionally, irrespective of the instrument, only a few notes would be simultaneously played at once. This suggests that C and W would be sparse, encouraging dimensionality reduction to benefit the efficiency of the algorithm in real-time.

The concepts described above suggest the development of a neural architecture capable of learning latent representations that are invariant to performance imperfections, yet retain the ability to accurately locate the encoded window (W') within the encoded context (C'). We define these representations as follows:

$$W' = E_w(W|\theta_w) \rightarrow \mathbb{R}^{e\times w}, \tag{3.1}$$

$$C' = E_c(C|\theta_c) \rightarrow \mathbb{R}^{e\times c}. \tag{3.2}$$

Here, θ_w and θ_c are the learnable parameters of the neural encoders E_w and E_c, respectively, with e indicating note dimension compression. Since our subsequent cross-correlation operations are performed along the time dimension, we do not compress the inputs along this dimension.

[2]While most practical systems cover only a subset of this range, we utilize the full set of MIDI note numbers to ensure compatibility with future transcription and accompaniment systems that may accommodate unconventional MIDI note ranges.

[3]While training the DNN, we supply batches of C and W at once. The batched equivalents are of the form $C \in \mathbb{R}^{B\times 128\times c}$ and $W \in \mathbb{R}^{B\times 128\times w}$, where B denotes the batch size used during the training phase.

E_w and E_c can be implemented using 1-D Convolutional Neural Network (CNN) layers with a kernel size of k and a stride of 1, applied after padding $k//2$ zeroes on both ends of the input[4]. To describe the underlying operations, consider an E_w architecture consisting of a 1-layer CNN such that $e = 64$ and $k = 3$, with $padding = 1$ on both ends and $stride = 1$, resulting in the CNN layer having 64 kernels of shape 128×3. This means that for piano-roll window event j, we obtain the element-wise product of weight i with the previous event ($W[:, j-1]$), the current event ($W[:, j]$), and the next event ($W[:, j+1]$), and sum the result to obtain the latent window element $W'[i, j]$. Thus, the encoders perform convolution along the time dimension, smoothing out potential imperfections persisting across neighboring piano-roll events for every window position. We also perform a similar operation within E_c for the context, thereby ensuring the downstream cross-correlation occurs within the same latent space.

3.2.2 Aligning the Solo with the Performance

Assuming the encoded window representation W' now exactly matches within the context representation C', score following translates into a classical template-matching task. The goal now is to find the most accurate location of the solo within the score. This is achieved using cross-correlation defined by the following equation:

$$P' = (C' \star W') \rightarrow \mathbb{R}^{1 \times (c+w-1)}, \tag{3.3}$$

where $\star$ denotes the cross-correlation operation. This is implemented by padding C' on both sides with a zero vector of size $e \times (w-1)$, and sliding W' across its discrete time dimension (represented by c). At each position, we compute the corresponding integer element of P' by summing the element-wise product of W' and the overlapping segment of C'. The elements of P' indicate the likelihood of the window's presence at that context position[5]. This setup aligns perfectly with the typical DL classification task. Consequently, the DNN model is trained using the cross-entropy loss:

$$L(P', Y) = \sum_{k=1}^{c+w-1} Y[k] \log(\sigma(P'[k])) + (1 - Y[k]) \log(1 - \sigma(P'[k])), \tag{3.4}$$

where σ denotes the softmax function, applied to the elements of P' indexed by k to derive template-match probabilities. $Y[k]$ represents the elements of the one-hot vector Y, indicating the ground truth window position within the context.

Now, we can train the model exclusively on $L(P', Y)$, aiming not just to predict window positions accurately but also to jointly-learn the imperfection-invariant, temporally-equivalent representations W' and C'. Also, this approach supports identifying the top-k most likely positions post training, laying the groundwork for heuristical score following, discussed in section 3.2.3.

Once the prescribed model has been trained, the most probable window position within the context, Y_{pred}, is trivially determined using the operation:

[4] "//" indicates integer division (*i.e.*, we drop the fractional part).

[5] Within the scope of this work, window position specifically refers to the position of its right edge; ie, a window position x would mean that it occupies score positions from $x - w$ to x

$$Y_{pred} = argmax(P') \tag{3.5}$$

Given the inherent unpredictability of performance imperfections, and to ensure that the accompaniment would consistently follow the performer, our approach treats score following and computer accompaniment as iterative processes, executed at predetermined intervals throughout a performance. This fixed rate of execution, f_e Hz, also enables executing the algorithm irrespective of the expected tempo of the solo, enabling us to support following a larger bandwidth of performance tempos.

3.2.3 Heuristic Score Following

To address the challenge of accurately predicting window positions in contexts with similar or repeating note sequences, our approach must extend beyond relying solely on eq. (3.5). While some existing solutions consider the entire solo performance up to the current time, this method significantly increases computational demand for longer performances. To mitigate this challenge while keeping c and w constant, we combine eq. (3.5) with a set of heuristic rules based on past predictions. We derive these rules by largely applying common-sense logic that is universally applicable to the flow of performances regardless of the instrument or genre involved.

The heuristic logic for determining the overall predicted score position includes the following steps:

1. When the performance consists of repeating patterns, cross-correlation might return similar probabilities for all of them. Therefore, beyond identifying the peak position with eq. (3.5), we also consider alternate positions having comparable probabilities. We find these by smoothing the DNN output vector, and identifying significant peaks in it using SciPy's `find_peaks` function[6] with a prominence threshold of at least 3. If no significant peaks are found, the position of the highest peak of the smoothed vector is selected.

2. At the start of the performance, the heuristic system might require some time to stabilize and accumulate performance data. During this phase, we only focus on the most prominent peak in the DNN's output, storing initial predictions in a ring buffer. Once sufficient data is accumulated, all significant peaks are evaluated in conjunction with recent data from this buffer.

3. Using linear regression on buffer data, we extrapolate a predicted position that is consistent with the performer's local tempo, serving as a reference to critique the DNN model's predictions. A model prediction is validated if it meets the following criteria:

   (a) It is greater or slightly less than the previous prediction.

   (b) It lies within a defined threshold relative to the buffer prediction.

   (c) The rate of change compared to the latest prediction is comparable to the buffer's trend.

[6] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

We derive the first rule by assuming that the performer is almost always going to perform monotonically from the start of the score to its end; the slight threshold of backtracking is to overcome from potential errors in preceding predictions. The second and third rules stem from the expectation that a performer would continue to maintain a rate similar to the most recent performance section (the section being under a second long).

4. If a valid model prediction is not found, we calculate the mean of the buffer and model predictions;

   (a) If this mean is close to the model's prediction, it is possible that both the buffer and model predictions might be off. In this case, a safe estimate would be to consider the mean value as the predicted position.

   (b) If not, the model prediction has jumped too far, and this estimation may be incorrect. Therefore, we use the buffer predicted value.

5. While relying on the buffer predicted position is useful to safeguard against isolated errors made by the model, it could inhibit quick response to changes in the performer's dynamics. Furthermore, errors from the linear interpolation could also accumulate. Thus, we place a limit on the possible number of consecutive buffer predictions. Upon reaching this limit, we assume that the model is responding to a change in the performance characteristics, and use its prediction as the estimated score location[7].

These heuristics add a practical layer to our score-following strategy, combining algorithmic estimation with an adaptability characteristic of human-like tracking, thus enhancing the reliability and responsiveness of the accompaniment system to the performer's nuances. The overall score following system is illustrated in fig. 3.1.

3.3 Implementation Details

This section delineates the methodologies employed in implementing the problem defined in section 3.2. We begin with discussing the architecture employed by the score following DNN, followed by the dataset and paradigm used for training it. Then, we discuss its integration into a comprehensive system designed for real-time performer tracking and low-latency accompaniment generation.

3.3.1 Model Architecture: Tyke

Building upon previously established insights of score progression, we hypothesize that the position of a new window is very likely to be present in the vicinity of the previous window location. This premise suggests that searching within a localized score segment—a smaller context—would be both time and compute efficient. However, significant imperfections in the performance could position the new window entirely outside this context, necessitating

[7] In this scenario, it would also be useful to perform the global scan approach, discussed in section 6.1.3.

searching within a larger section of the score with support from the previously predicted locations. Moreover, given the variable lengths of musical scores, our model must embody temporal-equivariance, as expounded in section 3.2.1.

Accordingly, we introduce **Tyke**, a compact convolutional model designed to pinpoint the window's location within a localized context. Tyke aims to determine the probabilities of the window belonging to each context position (refer to eqs. (3.3) to (3.5) and section 3.2.3). Tyke's design also allows for processing windows and contexts of arbitrary durations as long as they exceed its kernel dimensions, facilitating us to use Tyke variants having different values of c without additional training overhead. With this flexibility, we can even search within the entire score by simply setting c to the score length.

3.3.2 Training Paradigm

Dataset

The MAESTRO Dataset V3.0.0 [57], comprising approximately 200 hours of virtuosic piano performances with precise alignment ($\sim$ 3 ms) between MIDI notes and audio waveforms, forms the basis of our training data. It also includes a CSV file delineating the division of data into training, validation, and testing sets. To train Tyke, we exclusively utilize the MIDI files, processing them to create the **MAESTRO for Score Following - Static (MSF-S)** dataset as per algorithm 1. Once generated, the MSF-S samples consists of a tuple of C, W and Y. During training, we randomly provide these samples as input to Tyke.

MIDI Augmentations with MIDIOgre

To derive imperfection-invariant representations, as detailed in section 3.2.1, we developed **MIDIOgre**[8], a Python-based, MIT-licensed MIDI data augmentation toolkit. MIDIOgre can parse MIDI files as PrettyMIDI [58] objects to simulate performance imperfections on the fly, enhancing Tyke's training with a broad spectrum of potential imperfections. In our strategy, these imperfections are introduced randomly, without any grounding on human performers doing the same. This follows common applications of augmentation in the vision domain [42], adhering to the expectation that our model would be asymptotically exposed to the entire set of human-induced imperfections over multiple epochs of training. We hypothesize this not only ensures robustness against unpredictable performance variances but also minimizes overfitting.

We adhere to the common strategies described by MDTK [48] to implement the following augmentations in MIDIOgre:

1. `PitchShift`: Randomly transposes MIDI notes of randomly selected instruments.
2. `OnsetTimeShift`: Randomly alters note onset times while maintaining their durations.
3. `DurationShift`: Randomly modifies note durations while preserving their onset times.
4. `NoteDelete`: Randomly removes some notes from an instrument track.

[8] https://github.com/a-pillay/MIDIOgre

Algorithm 1 Create MSF-S

for each data split **do**
    Create a CSV to store the split data
    Initialize $n \leftarrow 0$
    while $n < n_{\text{split}}$ **do**
        Select a random MIDI file from the data split
        Process the piano instrument to a piano-roll
        for each note in piano-roll **do**
            if note velocity > 0 **then**
                note velocity $\leftarrow 1$
            else
                note velocity $\leftarrow 0$
            end if
        end for
        Determine a random position in the piano roll from where a context of duration c can be obtained
        Determine a random window start position
        if window is completely outside context **then**
            flag $\leftarrow$ True
        else
            flag $\leftarrow$ False
        end if
        Append to CSV: MIDI file name, context start, window start, flag
        $n \leftarrow n + 1$
    end while
end for

5. `NoteAdd`: Randomly inserts some notes into an instrument track.

For visualising how these augmentations appear within piano rolls, refer appendix A.

3.3.3 Accompaniment System Design[9]

In this section, we describe potential ways to synergistically combine existing auxillary components to form a comprehensive accompaniment system capable of listening to audio frames of the solo, convert them into piano-rolls, infer them using the score following DNN(s), and generate the accompaniment in response to the predicted position of the performer. In this regard, our overarching goal is a system having minimal latency while being sufficiently reactive to be able to track even the most rapidly-played, ephemeral notes.

Audio to MIDI Conversion via BasicPitch

Since providing an accompaniment system that should follow analog instruments means we would not readily have the solo in a digital format, the first component in our accompaniment system would involve converting the incoming solo audio frames into a binary piano-roll format that our score following DL model (see section 3.3.1) can process.

To this end, we could employ **BasicPitch** [59], a low-resource pre-trained Automatic Music Transcription (AMT) model that generalizes to a wide range of polyphonic instruments (including vocals). BasicPitch has been experimentally validated to have comparable performance to instrument-specific state-of-the-art AMT systems while being a considerably smaller model. Additionally, BasicPitch is also available as a Python library[10] that can be conveniently used to perform AMT on the provided audio through the command-line interface (CLI).

As a potential implementation, we first record a frame of the audio solo followed by processing it through suitable noise-filtering and gain conditioning DSP blocks. Subequently, we run it through BasicPitch to generate the corresponding piano-roll window that will be supplied to the downstream score following system. For maximizing efficiency, we could also omit BasicPitch's intermediate MIDI file generation process and finetune it to directly generate the window that can be processed as is by Tyke.

Parallel Processing

Considering that human auditory perception is sensitive to latencies greater than 10ms [60], we estimate a successful computer accompaniment system to have input-to-output latencies under 5ms. From a preliminary analysis, AMTs like BasicPitch and the score following model can be the main bottlenecks in our system. Additionally, Python's Global Interpreter Lock (GIL) places heavy restrictions on running all the system components in parallel.

As a workaround, we use the multithreading capabilities of QTThreadPool, available through PySide6[11], a Python wrapper for the QT6 graphics toolkit. Specifically, imple-

[9] Note that this subsection does not introduce new components but suggests ways to seamlessly integrate existing tools and techniques with the score follower discussed in section 3.3.1.

[10] https://github.com/spotify/basic-pitch

[11] https://pypi.org/project/PySide6/

menting dedicated threads for capturing the audio frames, executing the DNNs, updating the front end GUI and communicating the score position via Open Sound Control (OSC). Further, it would be beneficial to implement a shared tensor queue that gets populated by the latest recorded audio frame at a fixed time interval. Then, the DNN thread can fetch the latest data of size w from this queue and provide them as inputs to the model, at a rate of f_e per second.

GUI design

To visually validate the progression of score following, we suggest designing a simple graphical front end using PySide 6. The core elements displayed on the front end should include the score and the incoming notes from the solo. The current context, window and the detected window location must also be highlighted. Essentially, the front end serves largely as a qualitative evaluation tool, but can be extended upon to support functionalities as deemed necessary down the line.

Accompaniment using OSC and Digital Audio Workstation (DAW)

Once the window location within the score is identified, we must use this information to deliver the accompaniment. The OSC protocol[12] has been widely implemented as a means of exchanging musical information between different performance tools. It is natively supported by most popular DAWs and musical experimentation software like Max/MSP [13] and SuperCollider [14]. OSC messages can be communicated as text strings over the User Datagram Protocol (UDP) [61] protocol. Consequently, our accompaniment component could set up a UDP client over an available port, and periodically communicate the following information as OSC messages:

1. Estimated position of the solo on the score, as a time stamp.
2. Estimated tempo of the performer at the given instant (expressed in terms of its deviation from the base tempo set by the score).

We believe that any commonly available DAW can utilize the information described above to generate an appropriate accompaniment for the solo. A probable impediment in this method is that most DAWs implement their OSC Application Programming Interface (API) idiosyncratically. Thus, the system would have to communicate the messages in a format specific to the DAW being used. However, this added effort only manifests as an additional configuration effort that can be set once when the system is initialized.

Should future research determine that using DAWs is impractical for computer accompaniment, exploring alternative solutions like Dannenberg's *Accomplice* might be beneficial. Accomplice is an accompaniment system that provides fine control over tempo and score positions and includes functionalities to load, start, and stop projects, along with clock synchronization for precise timing.

[12] https://ccrma.stanford.edu/groups/osc/index.html
[13] https://cycling74.com/products/max
[14] https://supercollider.github.io/

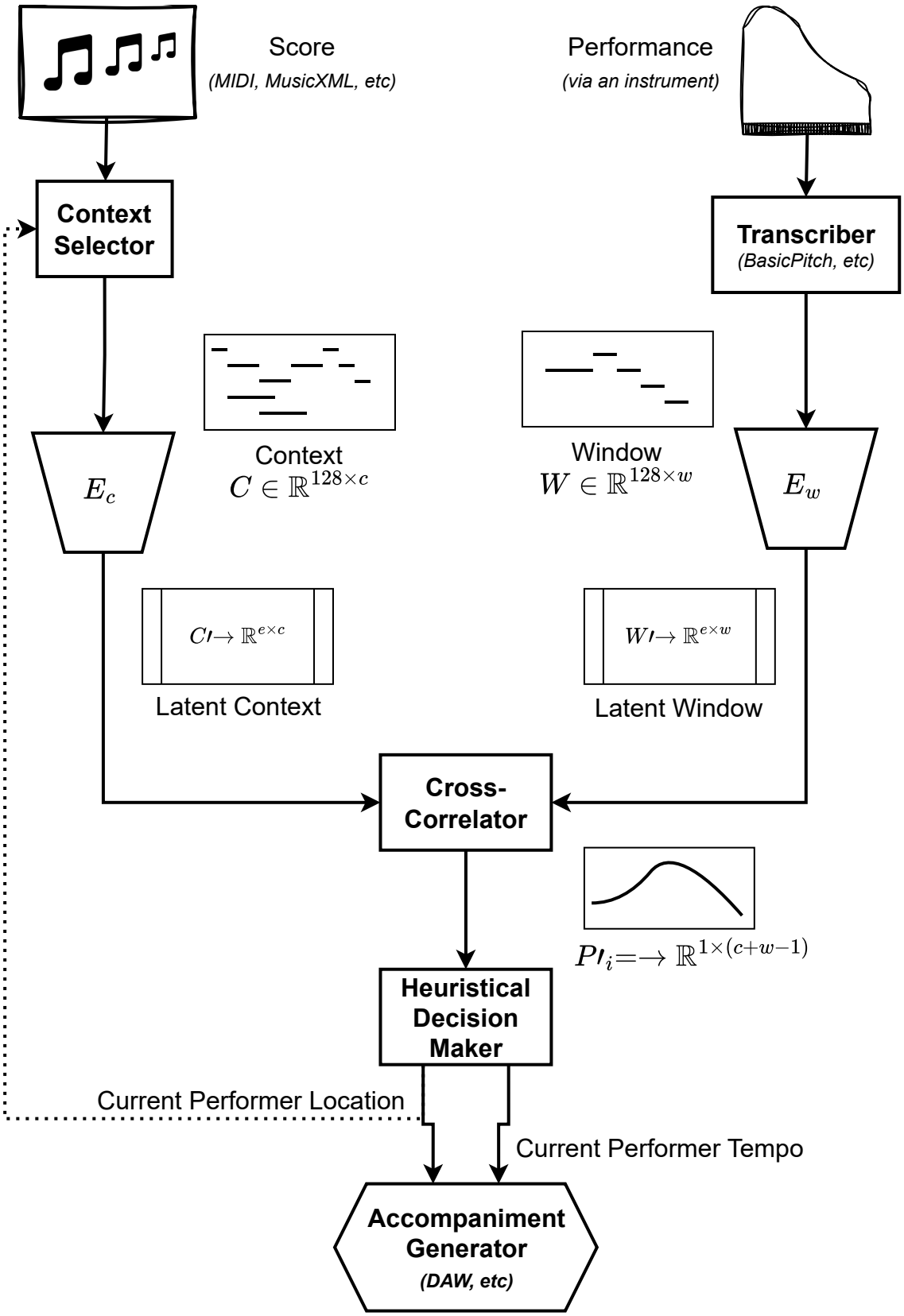

Figure 3.1: A block diagram of the score following system defined in section 3.2.

Chapter 4

Evaluation

In addition to the development of the system outlined in chapter 3, it is crucial to establish a set of metrics to evaluate its performance reliably, alongside further experimentation and refinement to achieve an optimal level of functionality. As discussed in section 2.5.4, we aim to consolidate and refine existing evaluation metrics to ensure their relevance and compatibility with both current and future research. This chapter begins with a definition of the metrics selected for our system's assessment, followed by establishing a set of experiments where we benchmark it on these metrics. In addition, we set thresholds of acceptable metric values, aiming to match or improve upon state-of-the-art score followers where comparable metrics are available. In scenarios lacking direct comparisons, we will deduce these thresholds based on general performance expectations from the system.

4.1 Training

In the absence of directly comparable neural frameworks for training Tyke, we utilize the cross-correlation between unencoded windows (W) and unencoded contexts (C) as described in section 3.2.2, to establish our *training baseline*[1]. This approach not only facilitates monitoring of Tyke's training progress and prevent overfitting but also aids in evaluating the effectiveness of the learned latent features in enhancing the score following accuracy of our system.

During the training phase, we monitor the following metrics:

1. **Cross-Entropy Loss** as a proxy to measure the model's classification error.
2. **Prediction Accuracy** to quantify the percentage of window locations Tyke correctly estimates within a 5ms range of their actual ground truth values.
3. **Baseline Prediction Accuracy** to determine the percentage of window locations accurately estimated by the cross-correlation of W and C within a 5-ms range of their actual ground truth values.

[1]This should not be confused with our score following baseline, Flippy, introduced in section 4.2.

Metric	Variable Name	
	Train Set	**Validation Set**
Cross-Entropy Loss	train_loss	val_loss
Prediction Accuracy	train_acc	val_acc
Baseline Prediction Accuracy	-	val_bacc

Table 4.1: List of training metrics for Tyke.

Table 4.1 lists the variables tracking these metrics across the train, validation, and test data splits during Tyke's training process. The following conditions serve as thresholds to ascertain successful training of Tyke:

1. To avoid overfitting: The ratio of validation to training accuracy ($\frac{val\_acc}{train\_acc}$) should be at least 0.75.

2. For accurate onset prediction: The validation accuracy (val_acc) should meet or exceed 0.9.

3. To outperform the baseline (piano-roll data cross-correlation): Validation accuracy (val_acc) should surpass the baseline prediction accuracy (val_bacc).

This setup aims to ensure that Tyke not only achieves high accuracy in score following but also surpasses existing methodologies in reliability and efficiency.

4.2 Inference Evaluation

Following the training of our DNN model, we assess its performance in score following by analyzing the model using prerecorded human performance data and corresponding scores. Our primary dataset is the (n)ASAP dataset by Peter et al. [62], chosen for its recency and the richness of its contents. (n)ASAP, derived from the ASAP dataset [63], consists of note-level annotations and is characterized by significant tempo variations and musical diversity. The original performances, sourced from the MAESTRO dataset [57], feature a broad spectrum of performer skill levels [33].

For evaluating our work, we select a set of metrics primarily adapted from Lee's 2022 work [33], which builds on the MIREX benchmark [55] and the methodologies of Thickstun et al. [64]. The metrics are defined as follows:

1. **Misalign Rate** (r_e): The percentage of performance events where predicted onset times deviate beyond a misalignment threshold, θ_e, from their expected values. We evaluate performances at thresholds $\theta_e \in \{25, 50, 75, 100, 125, 300, 500, 750, 1000\}$ ms.

2. **Alignment Error** (e_i): The absolute time difference between the predicted and actual onsets, reported as mean (μ_e) $\pm$ standard deviation (SD) (σ_e) over all aligned events.

3. **Latency** (l_i): The delay in recognizing an event by the score follower, expressed as mean (μ_l) $\pm$ SD (σ_l) for the entire performance.

As the *score following baseline*[2], we utilize the system from Lee's work, henceforth referred to as **Flippy** following the nomenclature used within its accompanying code repository[3]. Specifically, we employ Flippy in its "NSGT-CQT Online" mode, which features a Non Stationary Gabor Transform (NSGT) based feature extractor and an OLTW-based aligner, operating on frames of performance audio and the synthesized score. Our experiments adhere to the configuration Lee used to report their results, maintaining all system parameters at their default values[4], except for adjusting the "OLTW Weight to constrain the path for the i direction parameter" to 0.5.

While (n)ASAP contains note-level ground truth alignments between the MusicXML score and MIDI performance data, applying these directly to evaluate both systems is challenging. Thus, we devise the following evaluation methods:

1. Our system operates on a window of performance notes to determine its overall location within the score. While it is possible to indirectly determine the positions of the constituent notes within the window, this process is complex and could introduce errors orthogonal to the score following process. This is especially critical when predictions occur at score positions where no new notes are onset, as (n)ASAP contains only note-onset specific alignment data. Consequently, we derive ground truth predictions by performing unconstrained, offline DTW between the score and performance piano rolls using the dtaidistance[5] Python package. We then calculate the evaluation metrics as follows:

   (a) Alignment Error: For window i, performance window position X_i, ground truth score position Y_i, and predicted score position Y_{pred_i}:

   $$Y_i = dtw\_warping\_path(X_i) \tag{4.1}$$

   $$e_i = |Y_i - Y_{pred_i}| \tag{4.2}$$

   (b) Misalign Rate: Derived from e_i and θ_e as previously defined.

   (c) Latency: The difference in wall times measured from the recording of the last performance event to when the system generates a prediction output.

2. For Flippy, we adopt the string-matching based ground truth generation tool provided by the authors[6], for simplicity and consistency with reported results.

Given our DNN's training on the MAESTRO *train* split, it is critical to evaluate our system using (n)ASAP performances from the *test* split of MAESTRO exclusively. Of the 74 eligible performances, we were unable to generate Flippy ground truths for 26 due to the string-matching aligner exceeding Python's recursion limit of 5000. Additionally, 4 more

[2]The score following baseline differs from the training baseline described in section 4.1. The latter is utilized solely for monitoring Tyke's training progress.

[3]https://github.com/flippy-fyp/flippy

[4]https://github.com/flippy-fyp/flippy/blob/main/lib/args.py

[5]https://dtaidistance.readthedocs.io/en/latest/modules/dtw.html#dtaidistance.dtw.warping_paths_fast

[6]https://github.com/flippy-fyp/flippy-quantitative-testbench

performances were excluded since calculating their DTW warping paths exceeded our computational budget, leaving us with 44 (n)ASAP performances available for evaluation, detailed in table 4.2.

4.2.1 Inference Experiments

Our system relies on cross-correlation for rapid execution at a fixed frequency (f_e). However, this approach is sensitive to the base tempo of the performance. When the performance tempo significantly deviates from the tempo indicated by the score, as commonly observed in table 4.2, the resulting mismatch in the resolutions of the corresponding piano rolls can compromise the efficacy of the scale-sensitive cross-correlation algorithm. To understand the impact of these mismatches, we initially evaluate our system under ideal conditions by adjusting the score's base tempo to closely match the estimated tempo of the performance. We then analyze performance with the original score tempos to measure the effect of tempo mismatches. Additionally, we focus on a specific performance (P7) to explore how varying degrees of tempo mismatch influence our system's score following capability. This analysis involves plotting the misalignment rate (r_e) across a range of tempo mismatches.

Furthermore, we examine the robustness of our heuristic system by analyzing the influence of hyperparameter values on its performance. Specifically, we vary f_e while keeping all other hyperparameters constant, assessing the system's behavior on performance P8 at a fixed $\theta_e = 100$ ms.

4.3 Ablation Study

To thoroughly benchmark the MIDI data augmentation strategies described in section 3.3.2, we also conduct an ablation study. This study will compare the performance of our model with and without each specific augmentation technique to ascertain their individual contributions to the model's effectiveness. The study involves training multiple variations of the best performing Tyke architecture, each progressively disabling one of the augmentations detailed in section 3.3.2. We will also include a model variant trained entirely without augmentations for comparison. This series of ablations is designed to study the impact of each augmentation strategy on developing robust latent features that are essential for effective score following.

Performance Code	Performance	Duration (s)	Original Score Tempo (BPM)	Performance Tempo (BPM)
P1	Bach/Fugue/bwv_858/VuV01M	144.75	120	58.02
P2	Bach/Prelude/bwv_858/VuV01M	80.9	120	66.74
P3	Bach/Fugue/bwv_858/Zhang01M	129.19	120	65.01
P4	Bach/Prelude/bwv_858/Zhang01M	87.21	120	61.91
P5	Bach/Fugue/bwv_862/Song04M	155.01	120	54.18
P6	Bach/Prelude/bwv_862/Song04M	75.19	120	105.32
P7	Bach/Fugue/bwv_863/LeeN01M	169.17	120	58.16
P8	Bach/Prelude/bwv_863/LeeN01M	102.88	120	50.73
P9	Bach/Fugue/bwv_863/Shychko01M	194.03	120	50.71
P10	Bach/Prelude/bwv_863/Shychko01M	102.03	120	51.15
P11	Bach/Fugue/bwv_863/TongB01M	173.55	120	56.69
P12	Bach/Prelude/bwv_863/TongB01M	94.35	120	55.31
P13	Bach/Fugue/bwv_873/Lisiecki13M	122.01	120	103.63
P14	Bach/Prelude/bwv_873/Lisiecki13M	257.22	120	64.38
P15	Bach/Fugue/bwv_891/Duepree06M	218.25	120	166.59
P16	Bach/Prelude/bwv_891/Duepree06M	141.73	120	140.54
P17	Bach/Prelude/bwv_891/BLINOV04M	155.61	120	128
P18	Beethoven/Piano_Sonatas/10-1/Hou02M	278.94	98	85.87
P19	Beethoven/Piano_Sonatas/12-1/Kleisen05M	406.6	56	48.55
P20	Beethoven/Piano_Sonatas/16-1/BuiJL02M	281.44	132	138.04
P21	Beethoven/Piano_Sonatas/16-1/Khmara05M	283.75	132	136.91
P22	Beethoven/Piano_Sonatas/16-1/LeeSH02M	306.07	132	126.93
P23	Beethoven/Piano_Sonatas/16-1/LuoJ03M	264.95	132	146.63
P24	Beethoven/Piano_Sonatas/16-1/Woo05M	277.57	132	139.96
P25	Beethoven/Piano_Sonatas/18-1/ChenGuang03M	365.41	138	108.79
P26	Beethoven/Piano_Sonatas/18-1/LeungR02M	354.07	138	112.27
P27	Beethoven/Piano_Sonatas/18-1/Levitsky05M	352.44	138	112.79
P28	Beethoven/Piano_Sonatas/18-1/ZhangH05M	352.48	138	112.78
P29	Beethoven/Piano_Sonatas/5-1/Colafelice02M	272.96	197	186.18
P30	Beethoven/Piano_Sonatas/5-1/SunD02M	237.09	197	214.34
P31	Beethoven/Piano_Sonatas/8-3/Na06M	258.88	190	194.22
P32	Beethoven/Piano_Sonatas/9-1/Tysman05M	434.06	148	89.29
P33	Beethoven/Piano_Sonatas/9-3/Tysman05M	194.31	168	161.49
P34	Chopin/Etudes_op_10/12/Bult-ItoS04M	156.18	160	126.88
P35	Chopin/Etudes_op_10/12/HuNY03M	158.07	160	125.36
P36	Chopin/Etudes_op_10/12/LuoJ08M	136.2	160	145.5
P37	Chopin/Etudes_op_10/12/WuuE06M	150.65	160	131.54
P38	Chopin/Etudes_op_10/12/ZhangYunling02M	138.18	160	143.41
P39	Chopin/Sonata_2/4th/KaszoS16M	81.88	220	225.69
P40	Debussy/Pour_le_Piano/1/MunA12M	225.49	152	143.05
P41	Liszt/Concert_Etude_S145/1/Kleisen06M	220.05	120	105.79
P42	Liszt/Concert_Etude_S145/1/Woo06M	236.24	120	98.54
P43	Liszt/Concert_Etude_S145/2/Lu03M	155.91	240	213.97
P44	Rachmaninoff/Preludes_op_23/6/Nikiforov14M	170.9	72	60.38

Table 4.2: Shortlisted performances from the (n)ASAP dataset used for evaluating our system and the Flippy baseline. Observations from the two rightmost columns indicate a significant mismatch between the score-specified tempos and the actual performance tempos. As part of our evaluation experiments, we study the impact of these tempo mismatches on our system's score following capabilities.

4.4 Listening Evaluation

Beyond numerical metrics, we also perform listening tests on the performances listed in table 4.2 to evaluate the efficacy of our score follower from a holistic perspective. This allows us to assess the system's suitability for following performers in practice, identifying both strengths and areas needing improvement. We perform these tests by warping the score piano roll onto the performance using paths returned by the DTW ground truth and our score follower. We then synthesize these piano rolls using a DAW, and play the performance and the score through separate channels. Listening via separate ears helps us visualize how the evaluation metrics manifest in real life, and how well the system handles different kinds of performance deviations.

Chapter 5

Results

5.1 Training

With respect to the metrics defined in section 4.1, the **MiniTyke** architecture, as described in table 5.1, achieved optimal performance. This model incorporates a single 1D-CNN layer for E_c and E_w, followed by a Rectified Linear Unit (ReLU) activation layer. By setting e to 64, we compress the resultant latent representations C' and W' to half their original size, thereby simplifying downstream cross-correlation computations. MiniTyke is a remarkably compact DNN, with fewer than 50,000 parameters, thus incurring minimal computational costs during inference. We set the piano roll resolution to $\frac{1}{96}$s, with $c = 512$ and $w = 256$, corresponding to context durations of approximately 5.33s and window durations of approximately 2.67s, respectively. Training employed the AdamW optimizer with a weight decay of $1e-2$, a learning rate of $5e-4$, and the default values for β_1 and β_2. Additionally, a Cosine Annealing learning rate scheduler was utilized with a minimum learning rate of $1e-6$ and a quarter-cycle of 10 epochs. The training spanned 50 epochs, with each epoch comprising 500 training samples and 50 validation samples from the MSFS-S dataset described in section 3.3.2. To model performer imperfections, we applied MIDIOgre augmentations on W using the configurations outlined in table 5.2, identified following a qualitative analysis of human performances relative to their scores.

The batch size was set to 64, and the model was trained on a single NVIDIA GeForce RTX 3060 Mobile GPU over a duration of 67 minutes. Optimal results were achieved by the 45th epoch, as detailed in table 5.3. These results fulfill all objectives established in section 4.1, confirming the successful training of the MiniTyke model and its readiness for evaluation within the overall score following framework. Furthermore, the 6% improvement in validation accuracy over the training baseline cross-correlation algorithm indicates that the latent features learned by MiniTyke not only facilitate data compression but might also enhance the system's score following accuracy.

5.2 Inference Evaluation

After training the MiniTyke model, we integrate it with the heuristics described in section 3.2.3 to form a complete score following system, termed **HeurMiT**. We optimized

Layer (type)	Output Shape	Param #	Details
MiniTyke	[B, 767]	–	–
Sequential E_c*:*	[B, 64, 512]	–	–
Conv1d-1	[B, 64, 512]	24,640	kernel_size=3, stride=1, padding=1
ReLU-2	[B, 64, 512]	–	–
Sequential E_w*:*	[B, 64, 256]	–	–
Conv1d-3	[B, 64, 256]	24,640	kernel_size=3, stride=1, padding=1
ReLU-4	[B, 64, 256]	–	–
Total params		49,280	
Total mult-adds (M)		605.55	

Table 5.1: Detailed summary of the DNN architecture for MiniTyke for $c = 512$ and $w = 256$.

Augmentation	**Configuration**	**mode**	**Probability**
PitchShift	`max_shift = 5`	both	0.1
OnsetTimeShift	`max_shift = 0.5`	both	
DurationShift	`max_shift = 0.25`	both	
NoteDelete	-	-	
NoteDelete	`note_num_range`=(20, 120) `note_duration_range`=(0.5, 1.5) `restrict_to_instrument_time`=True	-	

Table 5.2: MIDIOgre augmentations used to train MiniTyke, with corresponding configurations used.

Model	**train_loss**	**val_loss**	**train_acc**	**val_acc**	**val_bacc**
MiniTyke	1.341	1.458	86%	94%	88%

Table 5.3: Best training performance for MiniTyke.

HeurMiT using recalibrated scores, achieving the best results with settings $f_e = 10$Hz, $w = 500$ (~ 5.21s), and $c = 1250$ (~ 13.02s). To smooth MiniTyke's output, we applied a moving average filter with a window size of 5 and set the ring buffer size to store the past 20 predictions, with the first 5 marking the stabilization phase. For a MiniTyke prediction to be considered valid, it must fall within -48 to +96 samples of the buffer predicted position, and the rate of change with respect to the last prediction must be between 0.5 and 1.5. The maximum number of consecutive buffer predictions allowed is set to 5. For evaluating Flippy, we adopted the default settings prescribed by Lee.

Utilizing the metrics outlined in section 4.2, we quantified HeurMiT's performance against the baseline system developed by Lee on valid instances from the (n)ASAP dataset. Our findings are presented in table 5.4. Flippy exhibited a better misalign rate (r_e) across all values of the misalignment threshold (θ_e). For alignment errors (e_i), HeurMiT performed slightly better than Flippy at $\theta_e < 125$ ms, but this trend reversed at higher values. These results

indicate that HeurMiT has more tendency to lose track of the performance, as evidenced by the higher values of r_e. In real-life, we want minimal e_i throughout the performance and in these situations, HeurMiT follows the performer more accurately than Flippy, provided it does not completely lose track. High values of e_i at larger θ_e could stem from HeurMiT not having safeguards against the window drifting completely outside the context due to accumulated errors in past predictions. In these cases, the cross-correlation might return an incorrect position with the highest likelihood[1], with heuristic rules attempting to establish a reasonable compromise.

A notable difference between the systems lies in their latency (l), with HeurMiT apparently outperforming Flippy by an order of 10^3. However, we refrain from claiming substantial gains in efficiency due to this disparity arising from the fundamentally different operational methodologies of the two systems. Flippy conducts OLTW between the spectral representations of the entire synthesized score and the available performance audio up to that point, which constitutes an audio-to-audio alignment algorithm with a complexity of $\mathcal{O}(\max(p, s))$, where p and s represent the lengths of the performance and the score, respectively. In contrast, HeurMiT utilizes lightweight cross-correlation on fixed-length piano-roll windows and contexts, yielding an $\mathcal{O}(1)$ algorithm. This is further enhanced by using Pytorch's `nn.functional.conv1d`[2], which includes computational optimizations for NVIDIA GPUs.

For a visual comparison of HeurMiT's predictions with the DTW ground truth for select performances, see appendix B.

Misalignment Threshold $[\theta_e]$ (ms)	Flippy (Lee [33])			HeurMiT (ours)		
	Misalign Rate $[r_e]$ (%)	Alignment Error $[e_i]$ (ms)	Latency $[l]$ (ms)	Misalign Rate $[r_e]$ (%)	Alignment Error $[e_i]$ (ms)	Latency $[l]$ (ms)
25	**83.12**	11.31 ± 12.3	1505.89 ± 509.99	92.99	**10.58 ± 7.28**	**1.1 ± 0.19 (cuda)** **6.07 ± 1.70 (cpu)**
50	**70.63**	22.47 ± 23.88	1547.46 ± 544.11	88.07	**19.94 ± 13.08**	
75	**60.82**	33.38 ± 33.27	1586.03 ± 596.42	80.97	**33.37 ± 21.52**	
100	**54.05**	43.69 ± 42.23	1678.77 ± 729.09	77.24	**40.91 ± 26.74**	
125	**49.66**	51.9 ± 53.54	1636.95 ± 781.11	74.11	**50.88 ± 31.34**	
300	**40.93**	**83.15 ± 90.84**	1459.31 ± 879.93	59.97	107.59 ± 77.35	
500	**37.06**	**122.83 ± 130.8**	1467.38 ± 942.84	54.35	159.67 ± 121.48	
750	**34.39**	**154.2 ± 171.99**	1445.23 ± 957.16	50.38	220.54 ± 177.89	
1000	**32.48**	**186 ± 215.02**	1441.31 ± 1005.59	47.18	284.34 ± 240.03	

Table 5.4: Comparison of score following inference evaluation metrics for HeurMiT vs. Flippy. For all metrics, lower values indicate better performance.

Table 5.5 presents the results when HeurMiT attempts to follow performances with tempo mismatches relative to the score. The extremely high values of r_e across all θ_e thresholds indicate that HeurMiT struggles to adequately follow these performances. Further analysis is illustrated in fig. 5.1, where we plot the impact of varying degrees of performance-score tempo mismatches for performance P7. We observe that HeurMiT's r_e exceeds beyond 50% as the mismatch exceeds ±5 BPM. This poor performance can be attributed to the non scale-invariant nature of cross-correlation and the small kernel size used in MiniTyke, which

[1] We discuss potential solutions to this issue in section 6.1.3.

[2] https://pytorch.org/docs/stable/generated/torch.nn.functional.conv1d.html

only considers 3 consecutive samples at a time. This critical limitation compromises the robustness of HeurMiT, which was intended to adapt to significant changes in performance tempo.

Misalignment Threshold [θ_e] (ms)	Misalign Rate [r_e] (%)	Alignment Error [e_i] (ms)
25	99	8.95 ± 5.95
50	98	19.92 ± 10.08
75	96	36.16 ± 20.43
100	95	45.19 ± 24.49
125	94	66.77 ± 28.41
300	89	145.27 ± 77.28
500	85	227.00 ± 131.01
750	82	331.53 ± 194.55
1000	78	435.14 ± 268.79

Table 5.5: Inference evaluation metrics for HeurMiT when there is a mismatch between the performance score tempos.

Fig. 5.2 shows how HeurMiT's misalign rate (r_e) varies with the inference frequency (f_e), ranging from 91.4% to 56.3% at $\theta_e = 100$ms. This variability indicates that HeurMiT's ability to follow performances is significantly influenced by the applied hyperparameters. Ideally, a robust score follower should not exhibit strong dependency on hyperparameter settings, as this necessitates adjustments for different pieces or performers, which is not favorable.

5.3 Ablation Study

For our ablation study on MIDIOgre augmentations, we trained MiniTyke models using the same configurations as defined in section 5.1, but over 25 epochs. The results of this study are presented in table 5.6. Across all ablation experiments, the values of r_e and e_i obtained are quite similar, and the comparative differences can be considered negligible. Consequently, the MIDI augmentations applied do not appear to offer any significant benefits to HeurMiT's robustness.

5.4 Listening Evaluation

In our listening evaluation, detailed observations were made on a subset of performances listed in table 4.2, providing insights into the practical quality of score following by HeurMiT. Notably, performances P7, P11, P21, P38, and P43 were followed effectively from start to finish. However, a perceivable lag was noted, particularly during fast note progressions, as in P11, P21, and P43. This lag, consistent across performances, could potentially be corrected by offsetting HeurMiT's predictions by the average lag observed.

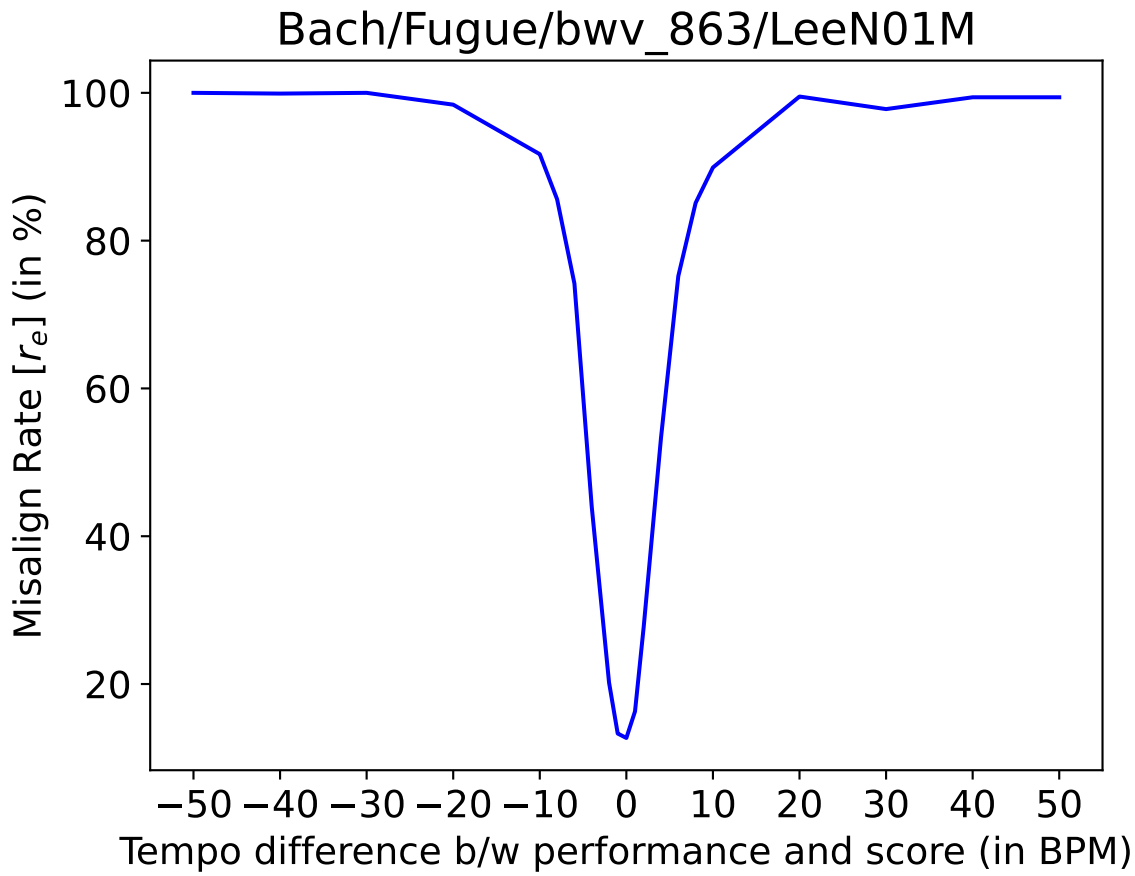

Figure 5.1: Impact of varying performance-score tempo mismatches on HeurMiT's misalign rate (r_e) for P7 at $\theta_e = 100$ms.

Ablation	Misalign Rate [r_e] (%)	Alignment Error [e_i] (ms)
All 5 Augmentations	76	43.92 ± 26.90
disabling `NoteAdd`	74	45.70 ± 28.07
disabling `NoteDelete`	76	45.04 ± 27.57
disabling `DurationShift`	76	43.58 ± 27.48
disabling `OnsetTimeShift`	76	43.87 ± 26.93
disabling `PitchShift` (*i.e.*, no augmentations applied)	74	44.33 ± 26.85

Table 5.6: Inference evaluation metrics for HeurMiT upon ablating the applied MIDIOgre augmentations during training. In the first experiment, we apply all five augmentations as described in table 5.2. Subsequent experiments progressively disable these augmentations; the second experiment omits `NoteAdd`, and this pattern continues until the final ablation, where all MIDIOgre augmentations are disabled.

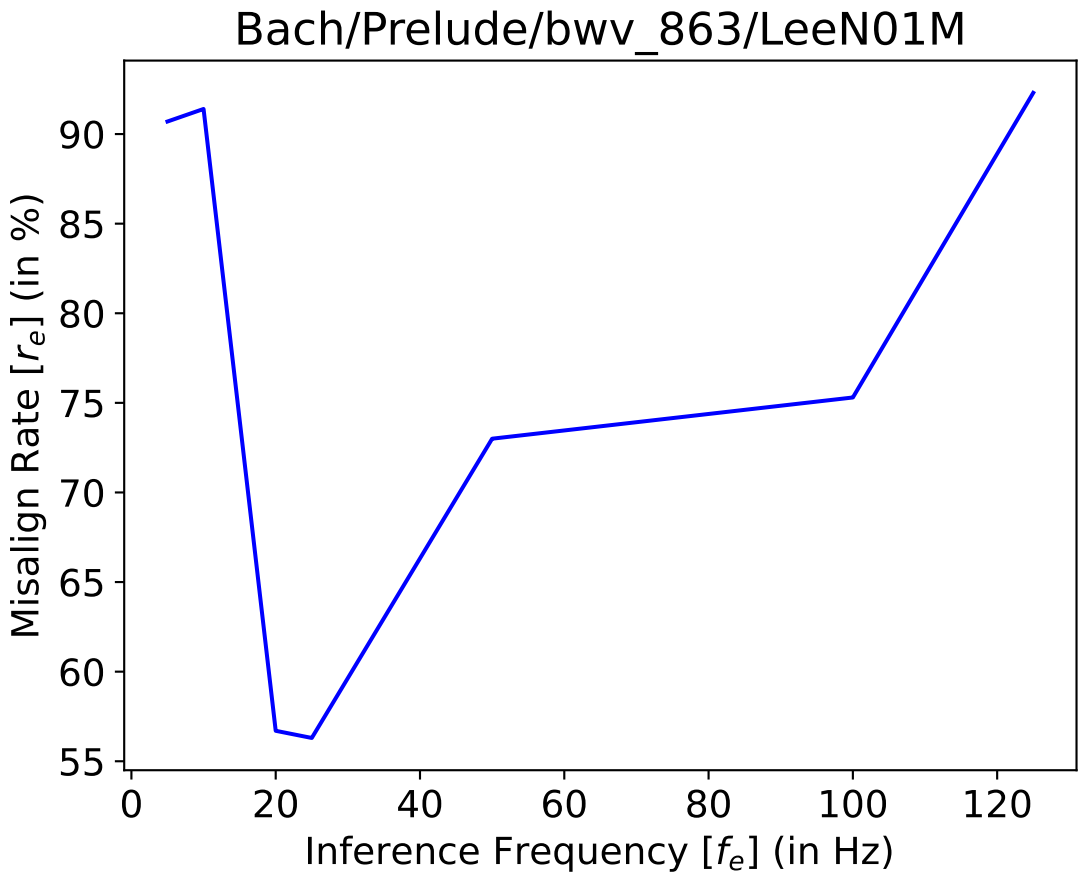

Figure 5.2: Impact of varying inference frequency (f_e) on HeurMiT's misalign rate (r_e) for P8 at $\theta_e = 100$ms.

Conversely, in cases like P25, P39, and P41, HeurMiT lost track almost immediately, likely because the heuristic system was still in its stabilization phase, lacking sufficient historical data to accurately assess the predictions from the cross-correlation algorithm. A potential solution could involve setting a limit on the maximum allowable increment in score predictions; however, further empirical testing is needed to confirm its effectiveness.

Despite the heuristic protections, there were instances where HeurMiT incorrectly estimated positions in pieces with repetitive note patterns.For P18, HeurMiT was thrown off seeing such patterns at two spots in the performance, but was able to recover quickly since they were followed by a complex note sequence in both cases. In P20, HeurMiT confused sections with similar progressions composed of slightly different notes. Similarly for P34, it jumped to another position in the vicinity of the ground truth having the exact same pattern.

5.5 Summary

Under ideal circumstances, evaluation results might suggest that HeurMiT performs comparably to the recently developed Flippy system by Lee. However, these findings are tempered by several limitations:

1. HeurMiT performs optimally when the performance tempo closely matches the score, or at a predetermined alternate rate. This is problematic in practice as performers often vary their tempo significantly. The lack of scale invariance in the cross-correlation method used by HeurMiT means that substantial tempo mismatches lead to poor performance.

2. Variations in hyperparameter settings significantly affect HeurMiT's performance, indicating a reliance on specific performance nuances. This reliance contradicts the goal of developing general-purpose heuristics and suggests a continuation of performance-dependent behaviors observed in other systems.

3. Unfortunately, our ablation studies do not reveal any distinct benefits from applying the MIDIOgre augmentations towards enhancing HeurMiT's robustness. This outcome, however, could be attributed to the generally subpar performance of our current system. We believe it would be beneficial to reperform our ablations with a better-performing system to more accurately assess the true contributions of the MIDI augmentations described in this work.

4. Even when inference metrics might suggest adequate performance, listening evaluations reveal that deviations as small as $\mu_e \approx 43$ms can still be perceptible and may influence performer behavior, potentially leading to a feedback loop of error accumulation.

These observations highlight the need for continued research into robust, efficient score following techniques that can operate independently of specific performance characteristics. Future work must explore alternative approaches to address the shortcomings identified within HeurMiT.

Chapter 6

Future Directions

Following the identified shortcomings of HeurMiT as discussed in section 5.5, this chapter outlines potential methods for addressing these issues. Initially, we will present straightforward enhancements to the current cross-correlation and heuristics-based approach. Subsequently, we explore alternative DL-based methodologies that may entirely circumvent the limitations of the template matching paradigm.

6.1 Improvements to the Current Approach

Cross-correlation is advantageous due to its fixed cost $\mathcal{O}(1)$ nature and its integration within a straightforward DL paradigm. We propose several enhancements to mitigate its inherent limitations and improve the robustness of the accompanying heuristics.

6.1.1 Cross-Correlation only on Note Onsets

As identified in section 5.4, mismatches in note durations between performances and scores are common. Although HeurMiT attempts to address this with MiniTyke trained on varied note durations using MIDIOgre, an alternative approach is to entirely disregard note durations and focus solely on their onsets. This method aligns with techniques used in symbol-matching systems where notes are represented as symbols, independent of their duration (section 2.1). By excluding note duration information, we aim to enhance the robustness of cross-correlation, especially when latent representations fail to adjust for duration mismatches. Additionally, this approach could slightly increase the tolerance for tempo variations, as it eliminates resolution discrepancies related to note offsets in the piano rolls. To implement this, we could truncate the note durations in the piano-roll to a single column width.

6.1.2 Multi-Scale Cross-Correlation

The scale-invariant limitations of cross-correlation were highlighted in section 5.5, particularly when facing significant tempo changes. Inspired by the Viola-Jones method in face detection [65], we consider employing a set of parallel multi-scale cross-correlation operations.

Each operation would apply a different resolution of the score piano roll on the performance piano roll. Leveraging MiniTyke's capability for batch tensor operations, this approach is anticipated to remain computationally feasible.

Upon determining the peak values across these operations, their magnitudes can serve as a confidence metric to select the most appropriate resolution and its corresponding position prediction. This method also permits integration of historical resolution data into heuristic confidence estimates, which could incorporate significant improvements but requires extensive empirical validation.

6.1.3 Global Search for Out-of-Context Performance Windows

A robust score follower should accommodate unexpected performance deviations that might lead to out-of-context windows, which current cross-correlation methods fail to indicate. This inability can cause prediction errors to accumulate rapidly, resulting in the system losing track of the performance. To mitigate this, we propose a *Global Search* method that recalibrates the context by searching the entire score length when the current window appears out-of-context. Although this method incurs an $\mathcal{O}(s)$ cost, it is intended to run infrequently, thus minimally adding to the overall computational cost.

Triggering this global search could be managed through heuristic rules after a certain number of consecutive buffer predictions or by incorporating a binary classifier within the Tyke framework. This classifier would operate on the cross-correlation vector P' to determine if the window is out-of-context, potentially using a simple fully-connected layer trained with Binary Cross-Entropy loss.

6.1.4 Incorporating Dynamic Heuristics

Instead of relying on static, performance-independent heuristics, it may be advantageous to explore methodologies that can learn and adapt to the dynamics of a performance. For example, our current system estimates the performer's tempo through linear regression on past predictions, which may fall short in situations where the performer frequently varies their tempo. In this direction, Xia et al. developed models that learn the interplay of musical expression between two collaborating performers [66], demonstrating that these systems outperform simple linear regression for mutual tempo estimation. Ideally, extending this adaptive approach to capture commonly observed performance dynamics between human soloists and accompanists could lead to the complete replacement of the rule-based heuristics described in section 3.2.3.

6.2 Exploring Alternate DL-Based Paradigms

Our current system exhibits sensitivity to tempo mismatches between the performance and score, and also relies heavily on heuristic rules. This section discusses potential DL paradigms that could overcome these limitations.

Research in chapter 2 highlights methods that follow notes as they come, and can inherently adjust to varying tempos. Exploring DNNs with Long Short-Term Memory (LSTM)

layers, trained on sequence-alignment prediction paradigms, offers a promising approach in this direction. Such networks could learn score embeddings and compare them against incoming performance notes to predict positions directly, potentially bypassing the need for heuristic adjustments. Incorporating attention mechanisms, either within LSTMs or through Transformers [51], could further enhance efficiency by focusing only on relevant historical data.

Implementing these methods requires a dataset that includes performances, scores, and precise note-level ground truth alignments. The (n)ASAP dataset, published subsequent to our initial approach, could be adapted for this purpose. However, the computational intensity of LSTMs and Transformers might necessitate compromises between real-time performance capabilities and system robustness during both training and inference.

6.3 Training Improvements

Enhancing the training data quality and diversity can also significantly improve the system's performance.

6.3.1 Further Exploration of MIDI Augmentations

Observations from section 5.4 indicate random deviations between performances and scores, which MIDIOgre can synthetically replicate during training. Future research could explore:

1. **Intensifying MIDIOgre Augmentations**: Current training uses a random transformation probability of 0.1 for MIDIOgre, based on qualitative analyses of MAESTRO performances (table 5.2). Quantitative research into the optimal configurations for MIDIOgre augmentations could refine our training approach.

2. **Extending MIDIOgre with New Augmentations**: Performers sometimes introduce note splits and trills not indicated in the score, leading to inconsistencies in HeurMiT's performance. Implementing `RandomNoteSplit` and `RandomTrill` functions could help the system better handle these variations. The former would randomly divide some notes into shorter segments, while the latter would need careful consideration of the typical trill characteristics around long notes.

6.3.2 Training On a Wider Variety of Performances

Although HeurMiT is designed to follow a broad range of musical styles and instruments, MiniTyke training has been confined to classical piano performances from the MAESTRO dataset. To truly enhance HeurMiT's versatility, it is crucial to expand the training datasets to include a variety of instruments and vocal performances. This would involve collecting new datasets that mirror the comprehensive nature of MAESTRO but encompass a wider range of musical expressions and styles. Ideally, these datasets would provide both audio and MIDI recordings along with accurate alignments, facilitating robust training and evaluation of score following systems across diverse musical categories.

Chapter 7

Conclusions

In this work, we introduced the challenges of score following and computer accompaniment as areas ripe for innovation. We reviewed existing methodologies, highlighting both their salient features and inherent limitations.

Following this review, we introduced a novel framework for score following, HeurMiT, which utilizes the capabilities of DNNs. This system is designed to interpret compressed latent feature representations from the score and performance, employing cross-correlation techniques enhanced with practical heuristics to accurately locate the performer's position within the score. To make these representations robust against the performer's deviations from the score, we developed a MIDI data augmentation toolkit and a template-matching based training paradigm, complete with a dataset derived from MAESTRO. Additionally, we outlined a comprehensive set of metrics and experiments for benchmarking our system against existing solutions and potential future developments.

Under ideal scenarios, HeurMiT delivers a score-following performance that is comparable to the existing Flippy baseline system developed by Lee [33], but is an $\mathcal{O}(1)$ system that is orders of magnitude more efficient in terms of computational time. However, subsequent experiments also revealed glaring limitations. Our system struggles with significant and unknown tempo deviations between the performance and the score. The underlying set of heuristics is sensitive to performance-specific nuances and requires manual adjustment for each performance to achieve optimal results. Further, listening evaluations on the performance audio and a warped version of the score derived from HeurMiT's predictions showed that the system tends to have a perceivable lag in following performances, often needing an offset adjustment. Moreover, HeurMiT tends to completely lose track of some performances early on during its stabilization phase, or gets confused among similar looking or repeating sequences of notes.

Following the analysis of these pitfalls, we discussed future research directions that either seek to enhance the template-matching paradigm or explore alternate approaches using DL that are independent of the tempo sensitivity issues observed with HeurMiT.

In conclusion, while our work in its present form is not yet ready for real-world application in score following, it represents a significant exploration into the potential of DL-based neural score following systems. By presenting our methodology and detailing both the strengths and limitations of our work, we aim to inspire future research towards identifying more robust and efficient methods for tracking performances across a diverse array of instruments,

genres, and styles using DNNs. We view the challenge of score following and computer accompaniment not merely as a technical hurdle, but as an opportunity to fundamentally enhance musical collaboration, learning, and creativity through the power of computers. By effectively addressing these challenges, we aim to empower musicians and learners alike, broadening the scope of musical expression and collaboration.

Appendix A

Visualizing MIDIOgre Augmentations

This chapter provides a visual representation of the effects of MIDIOgre augmentations within the piano roll domain. To illustrate these augmentations, we selected an excerpt from Beethoven's Bagatelle No. 25, commonly known as *Für Elise*. The following plots will demonstrate how various MIDIOgre transformations alter the original musical piece, providing insights into how these augmentations might influence the training of our deep learning models by simulating diverse performance variations.

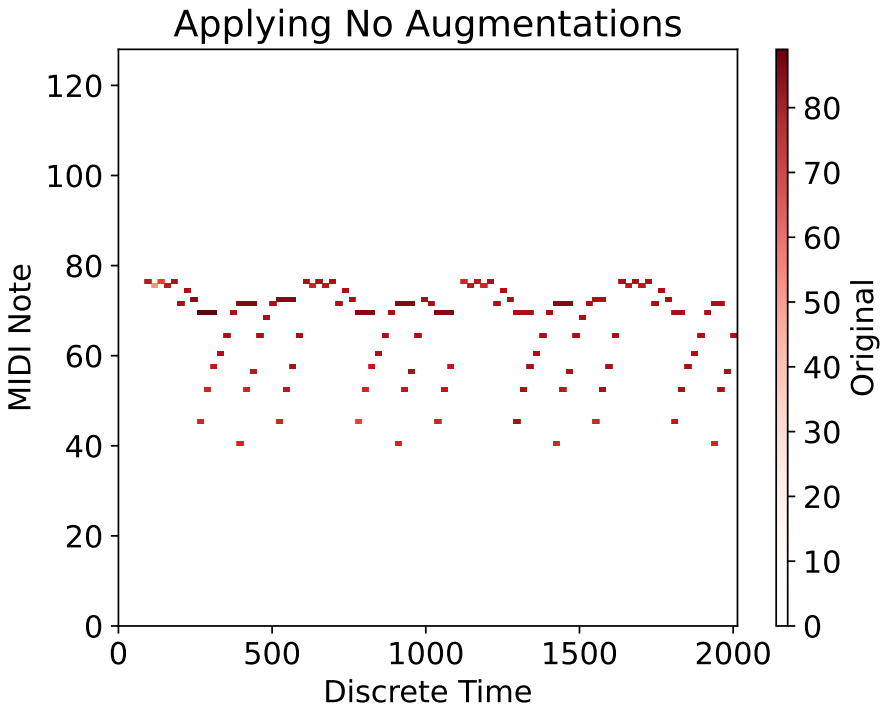

Figure A.1: Original performance without any MIDIOgre augmentations applied.

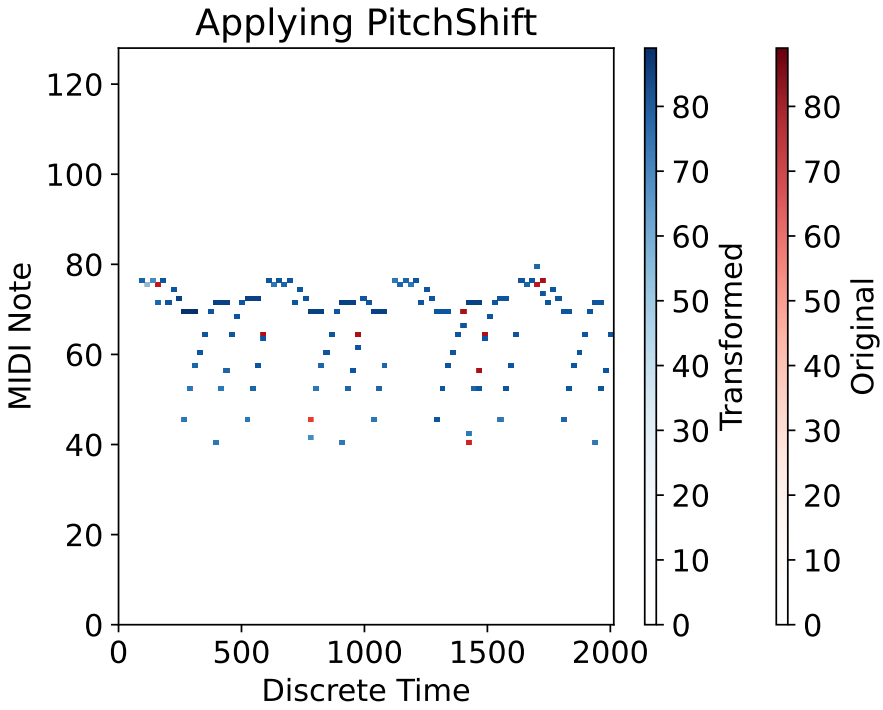

Figure A.2: Original performance vs PitchShift transforation; `PitchShift(max_shift=5, mode='both', p=0.1)`

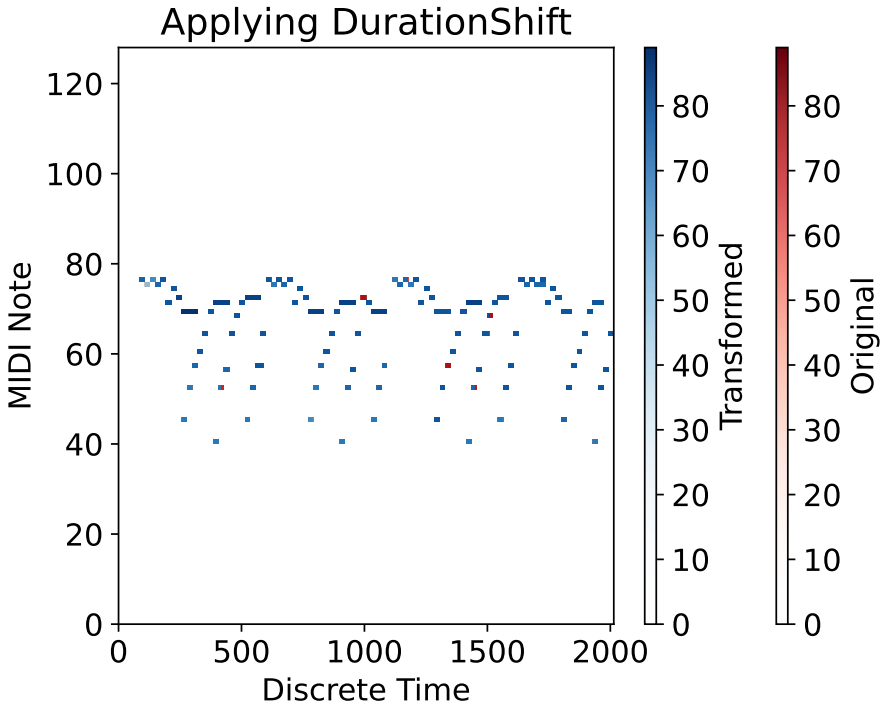

Figure A.3: `DurationShift(max_shift=0.25, mode='both', p=0.1)`

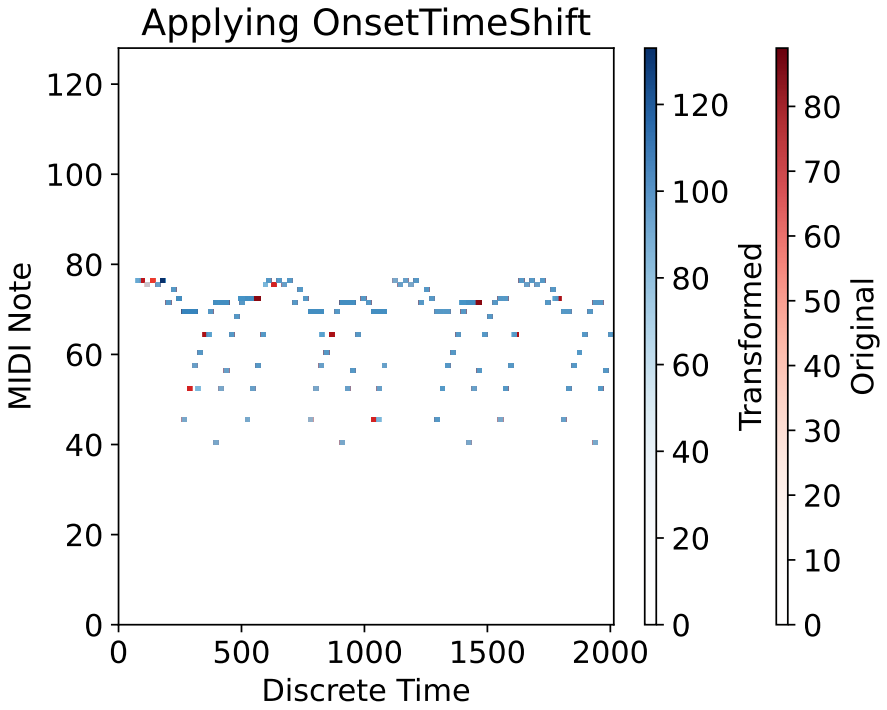

Figure A.4: `OnsetTimeShift(max_shift=0.5, mode='both', p=0.1)`

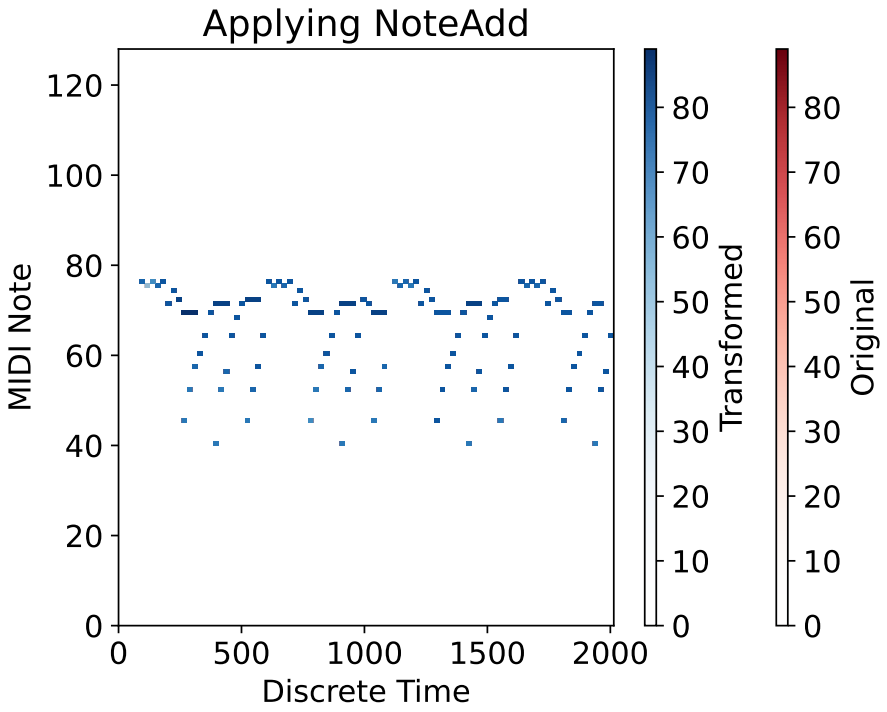

Figure A.5: `NoteAdd(note_num_range=(25, 120), note_duration_range=(0.5, 1.5), restrict_to_instrument_time=True, p=0.1),`

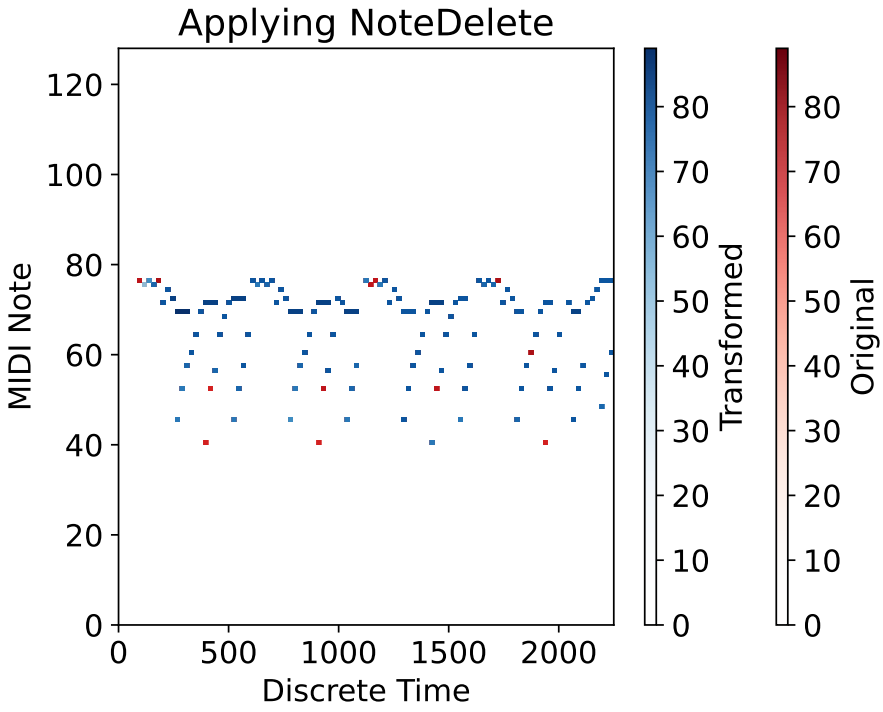

Figure A.6: `NoteDelete(p=0.1)`

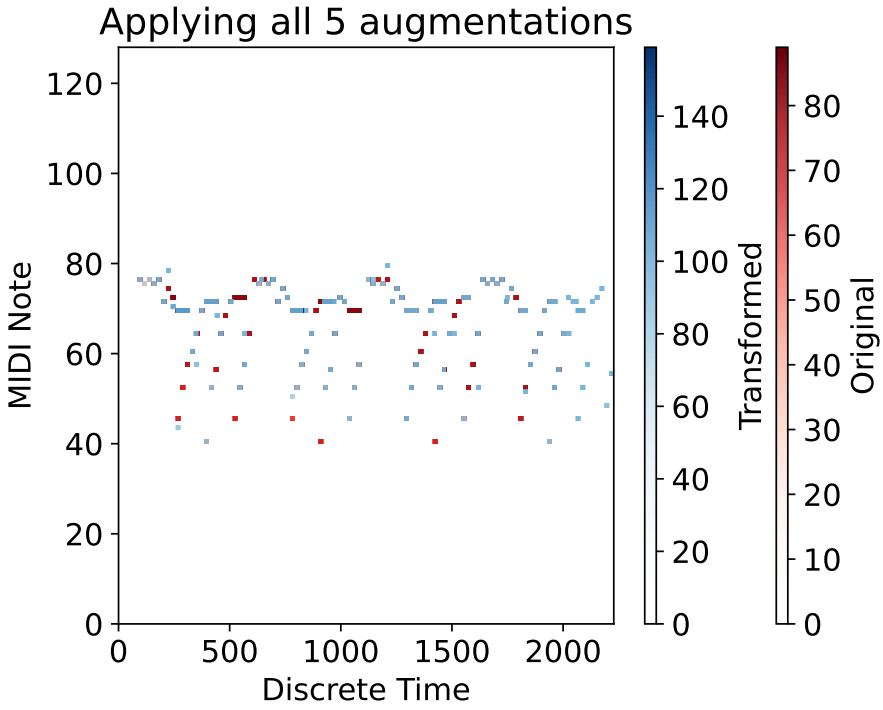

Figure A.7: All 5 augmentations applied together.

Appendix B

Tyke Inference Evaluation Plots

This chapter presents visualizations that illustrate the performance of HeurMiT's score following capabilities, displayed in blue, in comparison to the ground truth DTW warping paths, shown in green, for selected (n)ASAP performances. For a detailed explanation of the methodologies used to compute these metrics, please refer to section 4.2. Further analysis of these observations is available in section 5.2. Each plot is labeled with the corresponding performance code as listed in table 4.2.

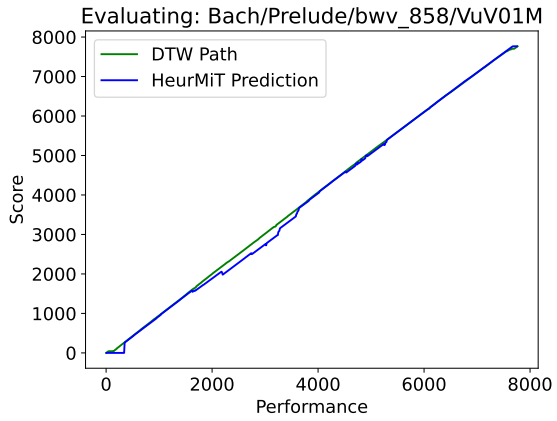

Figure B.1: P2

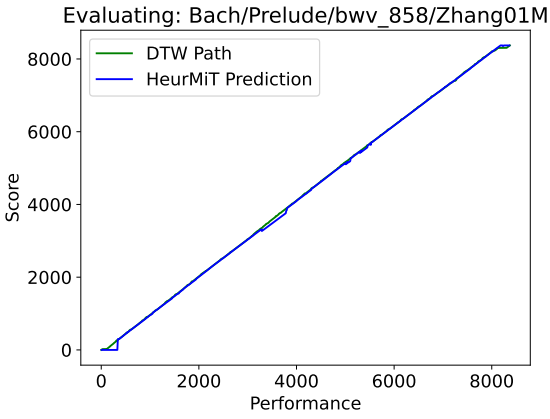

Figure B.2: P4

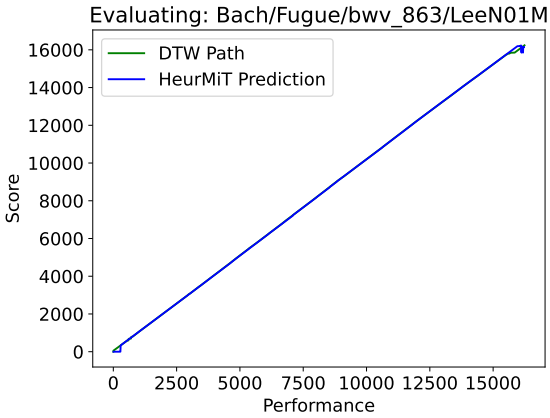

Figure B.3: P7

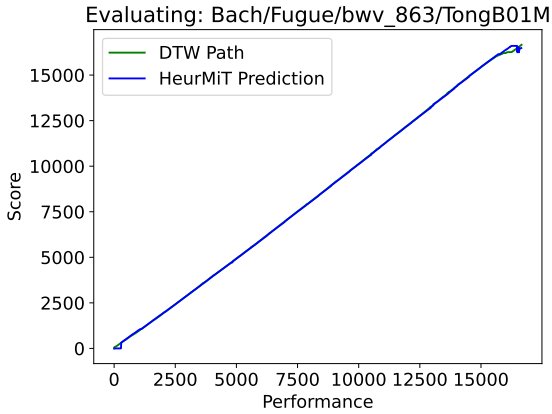

Figure B.4: P11

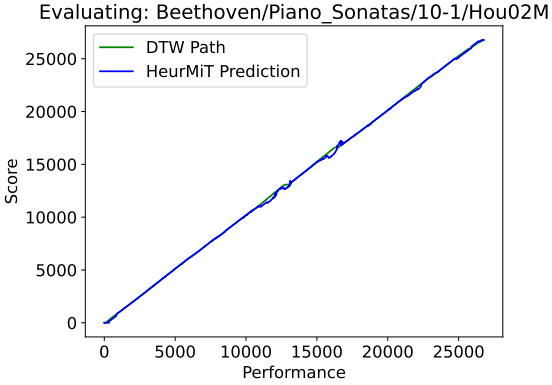

Figure B.5: P18

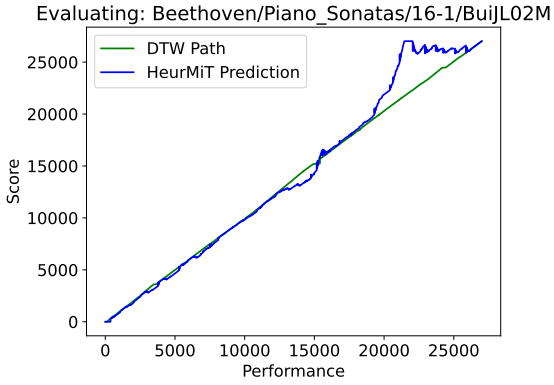

Figure B.6: P20

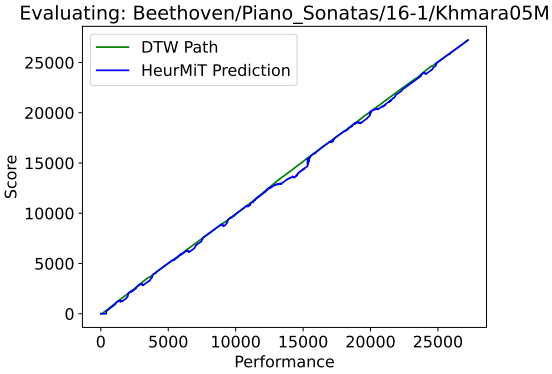

Figure B.7: P21

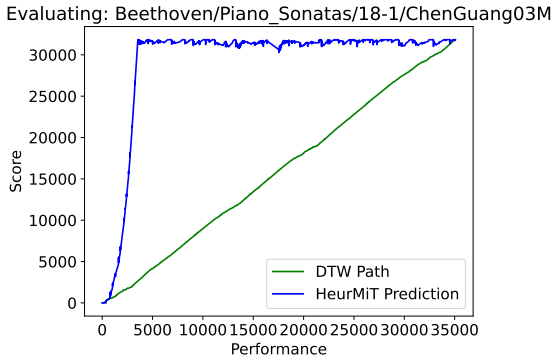

Figure B.8: P25

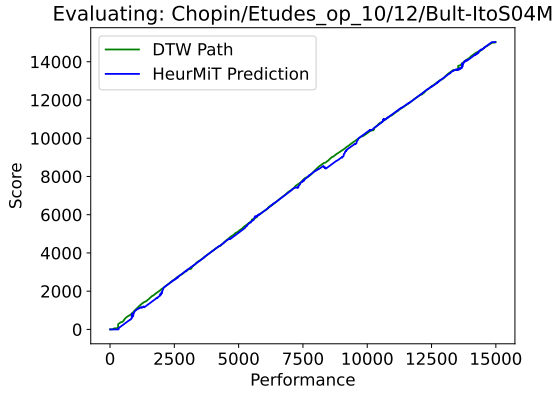

Figure B.9: P34

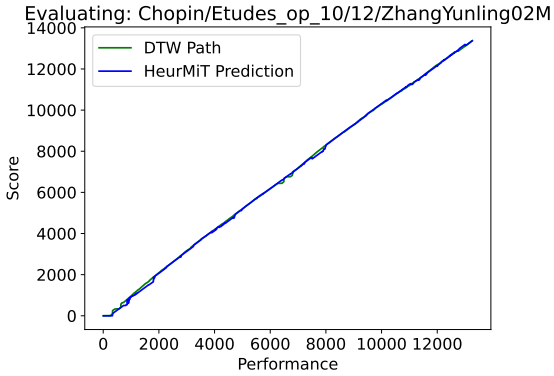

Figure B.10: P38

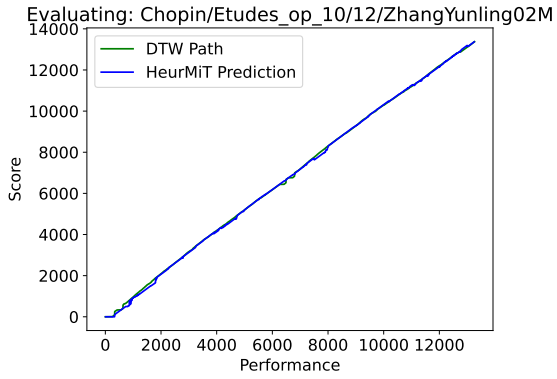

Figure B.11: P39

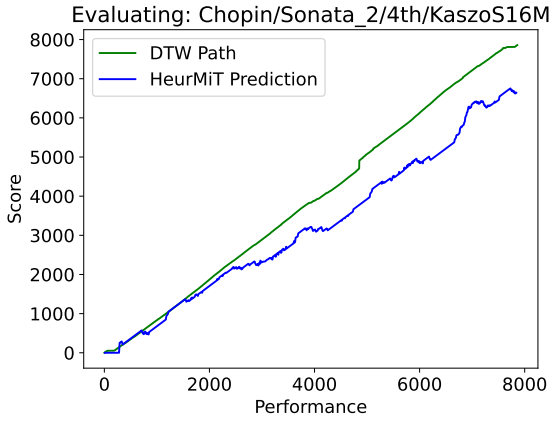

Figure B.12: P39

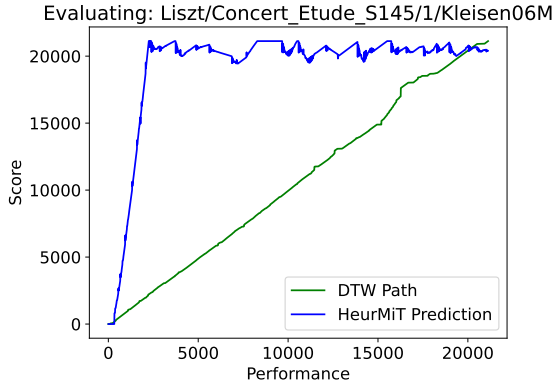

Figure B.13: P41

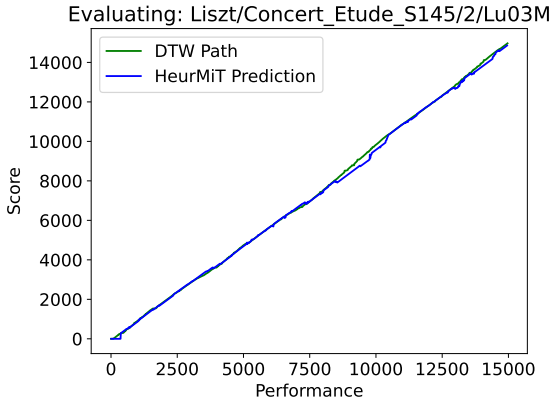

Figure B.14: P43

References

[1] R. B. Dannenberg, "An on-line algorithm for real-time accompaniment," in *International Conference on Mathematics and Computing*, 1984. URL: https://api.semanticscholar.org/CorpusID:9472796.

[2] H. Touvron, L. Martin, K. R. Stone, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *ArXiv*, vol. abs/2307.09288, 2023. URL: https://api.semanticscholar.org/CorpusID:259950998.

[3] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *ArXiv*, vol. abs/2212.04356, 2022. URL: https://api.semanticscholar.org/CorpusID:252923993.

[4] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," *ArXiv*, vol. abs/2102.12092, 2021. URL: https://api.semanticscholar.org/CorpusID:232035663.

[5] B. Vercoe, "The synthetic performer in the context of live performance," in *International Conference on Mathematics and Computing*, 1984. URL: https://api.semanticscholar.org/CorpusID:35821593.

[6] J. J. Bloch and R. B. Dannenberg, "Real-time computer accompaniment of keyboard performances," in *International Conference on Mathematics and Computing*, 1985. URL: https://api.semanticscholar.org/CorpusID:45811997.

[7] R. B. Dannenberg and H. Mukaino, "New techniques for enhanced quality of computer accompaniment," in *International Conference on Mathematics and Computing*, 1988. URL: https://api.semanticscholar.org/CorpusID:6249982.

[8] B. Vercoe and M. Puckette, "Synthetic rehearsal: Training the synthetic performer," in *International Conference on Mathematics and Computing*, 1985. URL: https://api.semanticscholar.org/CorpusID:44464828.

[9] B. Baird, D. Blevins, and N. Zahler, "Artificial intelligence and music: Implementing an interactive computer performer," *Computer Music Journal*, vol. 17, p. 73, 1993. URL: https://api.semanticscholar.org/CorpusID:61718818.

[10] M. Puckette and C. Lippe, "Score following in practice," in *International Conference on Mathematics and Computing*, 1992. URL: https://api.semanticscholar.org/CorpusID:45852326.

[11] J. Vantomme, "Score following by temporal pattern," *Computer Music Journal*, vol. 19, p. 50, 1995. URL: https://api.semanticscholar.org/CorpusID:61646605.

[12] L. Grubb and R. B. Dannenberg, "A stochastic method of tracking a vocal performer," in *International Conference on Mathematics and Computing*, 1997. URL: https://api.semanticscholar.org/CorpusID:32643313.

[13] B. Pardo and W. P. Birmingham, "Improved score following for acoustic performances," in *International Conference on Mathematics and Computing*, 2002. URL: https://api.semanticscholar.org/CorpusID:12104974.

[14] C. Raphael, "Automatic segmentation of acoustic musical signals using hidden markov models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, pp. 360–370, 1999. URL: https://api.semanticscholar.org/CorpusID:17597188.

[15] N. Orio and F. Déchelle, "Score following using spectral analysis and hidden markov models," in *International Conference on Mathematics and Computing*, 2001. URL: https://api.semanticscholar.org/CorpusID:14429211.

[16] D. Schwarz, N. Orio, and N. Schnell, "Robust polyphonic midi score following with hidden markov models," in *International Conference on Mathematics and Computing*, 2004. URL: https://api.semanticscholar.org/CorpusID:964885.

[17] B. Pardo and W. P. Birmingham, "Modeling form for on-line following of musical performances," in *AAAI Conference on Artificial Intelligence*, 2005. URL: https://api.semanticscholar.org/CorpusID:2463549.

[18] A. Cont and P. Vi, "Improvement of observation modeling for score following," 2004. URL: https://api.semanticscholar.org/CorpusID:9210006.

[19] A. Cont, "Realtime audio to score alignment for polyphonic music instruments, using sparse non-negative constraints and hierarchical hmms," *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, pp. V–V, 2006. URL: https://api.semanticscholar.org/CorpusID:6532374.

[20] A. Cont, "Antescofo: Anticipatory synchronization and control of interactive parameters in computer music," in *International Conference on Mathematics and Computing*, 2008. URL: https://api.semanticscholar.org/CorpusID:16408979.

[21] P. Cuvillier and A. Cont, "Coherent time modeling of semi-markov models with application to real-time audio-to-score alignment," *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2014. URL: https://api.semanticscholar.org/CorpusID:17854521.

[22] N. Montecchio and N. Orio, "Automatic alignment of music performances with scores aimed at educational applications," *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pp. 17–24, 2008. URL: https://api.semanticscholar.org/CorpusID:24362375.

[23] E. Nakamura, T. Nakamura, Y. Saito, N. Ono, and S. Sagayama, "Outer-product hidden markov model and polyphonic midi score following," *Journal of New Music Research*, vol. 43, pp. 183–201, 2014. URL: https://api.semanticscholar.org/CorpusID:17339720.

[24] S. Sagayama, T. Nakamura, E. Nakamura, Y. Saito, H. Kameoka, and N. Ono, "Automatic music accompaniment allowing errors and arbitrary repeats and jumps," *Journal of the Acoustical Society of America*, vol. 21, p. 035 003, 2014. URL: https://api.semanticscholar.org/CorpusID:123547804.

[25] M. Hori, C. M. Wilk, and S. Sagayama, "Piano practice evaluation and visualization by hmm for arbitrary jumps and mistakes," *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–5, 2019. URL: https://api.semanticscholar.org/CorpusID:122334670.

[26] C. E. C. Chacón, S. D. Peter, P. Hu, E. Karystinaios, F. Henkel, F. Foscarin, N. Varga, and G. Widmer, "The accompanion: Combining reactivity, robustness, and musical expressivity in an automatic piano accompanist," *ArXiv*, vol. abs/2304.12939, 2023. URL: https://api.semanticscholar.org/CorpusID:258309692.

[27] S. Dixon, "Live tracking of musical performances using on-line time warping," 2005. URL: https://api.semanticscholar.org/CorpusID:7356231.

[28] S. Dixon and G. Widmer, "Match: A music alignment tool chest," in *International Society for Music Information Retrieval Conference*, 2005. URL: https://api.semanticscholar.org/CorpusID:1748164.

[29] A. Arzt, G. Widmer, and S. Dixon, "Automatic page turning for musicians via real-time machine listening," in *European Conference on Artificial Intelligence*, 2008. URL: https://api.semanticscholar.org/CorpusID:14752757.

[30] F. J. Rodríguez-Serrano, J. J. Carabias-Orti, P. Vera-Candeas, and D. Martínez-Muñoz, "Tempo driven audio-to-score alignment using spectral decomposition and online dynamic time warping," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, pp. 1–20, 2016. URL: https://api.semanticscholar.org/CorpusID:2816862.

[31] K. Suzuki, Y. Ueda, S. A. Raczyński, N. Ono, and S. Sagayama, "Real-time audio to score alignment using locally-constrained dynamic time warping of chromagrams," 2010. URL: https://api.semanticscholar.org/CorpusID:18148894.

[32] J. J. Carabias-Orti, F. J. Rodríguez-Serrano, P. Vera-Candeas, N. Ruiz-Reyes, and F. J. Canadas-Quesada, "An audio to score alignment framework using spectral factorization and dynamic time warping," in *International Society for Music Information Retrieval Conference*, 2015. URL: https://api.semanticscholar.org/CorpusID:18255173.

[33] L. H. Lee, "Musical score following and audio alignment," *ArXiv*, vol. abs/2205.03247, 2022. URL: https://api.semanticscholar.org/CorpusID:248562558.

[34] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," 2015. URL: https://api.semanticscholar.org/CorpusID:3074096.

[35] K. Hornik, M. B. Stinchcombe, and H. L. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989. URL: https://api.semanticscholar.org/CorpusID:2757547.

[36] D. Kim, C.-H. Lai, W.-H. Liao, N. Murata, Y. Takida, T. Uesaka, Y. He, Y. Mitsufuji, and S. Ermon, "Consistency trajectory models: Learning probability flow ode trajectory of diffusion," *ArXiv*, vol. abs/2310.02279, 2023. URL: https://api.semanticscholar.org/CorpusID:263622294.

[37] S. Rouard, F. Massa, and A. D'efossez, "Hybrid transformers for music source separation," *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2022. URL: https://api.semanticscholar.org/CorpusID:253553270.

[38] A. Gupta, L. Yu, K. Sohn, X. Gu, M. Hahn, F.-F. Li, I. Essa, L. Jiang, and J. Lezama, "Photorealistic video generation with diffusion models," *ArXiv*, vol. abs/2312.06662, 2023. URL: https://api.semanticscholar.org/CorpusID:266163109.

[39] H. Lu, W. Liu, B. Zhang, *et al.*, "Deepseek-vl: Towards real-world vision-language understanding," 2024. URL: https://api.semanticscholar.org/CorpusID:268297008.

[40] J. Liu, C. Li, Y. Ren, F. Chen, and Z. Zhao, "Diffsinger: Singing voice synthesis via shallow diffusion mechanism," in *AAAI Conference on Artificial Intelligence*, 2021. URL: https://api.semanticscholar.org/CorpusID:235262772.

[41] R. Li, L. B. Allal, Y. Zi, *et al.*, "Starcoder: May the source be with you!" *ArXiv*, vol. abs/2305.06161, 2023. URL: https://api.semanticscholar.org/CorpusID:258588247.

[42] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, pp. 1–48, 2019. URL: https://api.semanticscholar.org/CorpusID:195811894.

[43] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. J. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6022–6031, 2019. URL: https://api.semanticscholar.org/CorpusID:152282661.

[44] S. Wei, S. Zou, F. Liao, and W. Lang, "A comparison on data augmentation methods based on deep learning for audio classification," *Journal of Physics: Conference Series*, vol. 1453, 2020. URL: https://api.semanticscholar.org/CorpusID:215994154.

[45] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Interspeech*, 2019. URL: https://api.semanticscholar.org/CorpusID:121321299.

[46] J. Thickstun, D. L. W. Hall, C. Donahue, and P. Liang, "Anticipatory music transformer," *ArXiv*, vol. abs/2306.08620, 2023. URL: https://api.semanticscholar.org/CorpusID:259164755.

[47] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "Musicbert: Symbolic music understanding with large-scale pre-training," *ArXiv*, vol. abs/2106.05630, 2021. URL: https://api.semanticscholar.org/CorpusID:235391043.

[48] A. Mcleod, J. Owers, and K. Yoshii, "The midi degradation toolkit: Symbolic music augmentation and correction," in *International Society for Music Information Retrieval Conference*, 2020. URL: https://api.semanticscholar.org/CorpusID:222090453.

[49] N. Jiang, S. Jin, Z. Duan, and C. Zhang, "Rl-duet: Online music accompaniment generation using deep reinforcement learning," in *AAAI Conference on Artificial Intelligence*, 2020. URL: https://api.semanticscholar.org/CorpusID:211069124.

[50] Z. Wang, K. Zhang, Y. Wang, *et al.*, "Songdriver: Real-time music accompaniment generation without logical latency nor exposure bias," *Proceedings of the 30th ACM International Conference on Multimedia*, 2022. URL: https://api.semanticscholar.org/CorpusID:252211925.

[51] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Neural Information Processing Systems*, 2017. URL: https://api.semanticscholar.org/CorpusID:13756489.

[52] S.-L. Wu, C. Donahue, S. Watanabe, and N. J. Bryan, "Music controlnet: Multiple time-varying controls for music generation," *ArXiv*, vol. abs/2311.07069, 2023. URL: https://api.semanticscholar.org/CorpusID:265150356.

[53] S. D. Peter, "Online symbolic music alignment with offline reinforcement learning," *ArXiv*, vol. abs/2401.00466, 2023. URL: https://api.semanticscholar.org/CorpusID:266349682.

[54] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014. URL: https://api.semanticscholar.org/CorpusID:11212020.

[55] A. Cont, D. Schwarz, N. Schnell, and C. Raphael, "Evaluation of real-time audio-to-score alignment," in *International Society for Music Information Retrieval Conference*, 2007. URL: https://api.semanticscholar.org/CorpusID:9901746.

[56] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *ArXiv*, vol. abs/1912.01703, 2019. URL: https://api.semanticscholar.org/CorpusID:202786778.

[57] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," *ArXiv*, vol. abs/1810.12247, 2018. URL: https://api.semanticscholar.org/CorpusID:53094405.

[58] C. Raffel and D. P. W. Ellis, "Intuitive analysis, creation and manipulation of midi data with pretty-midi," in *International Conference on Music Information Retrieval Late Breaking and Demo Papers*, 2014. URL: https://colinraffel.com/publications/ismir2014intuitive.pdf.

[59] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, "A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation," *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 781–785, 2022. URL: https://api.semanticscholar.org/CorpusID:247595162.

[60] D. Lelic, G. Stiefenhofer, E. Lundorff, and T. Neher, "Hearing aid delay in open-fit devices: Preferred sound quality in listeners with normal and impaired hearing.," *JASA express letters*, vol. 2 10, p. 104 803, 2022. URL: https://api.semanticscholar.org/CorpusID:253257633.

[61] J. Postel, "User datagram protocol," *RFC*, vol. 768, pp. 1–3, 1980. URL: https://api.semanticscholar.org/CorpusID:32952415.

[62] S. D. Peter, C. E. C. Chacón, F. Foscarin, A. Mcleod, F. Henkel, E. Karystinaios, and G. Widmer, "Automatic note-level score-to-performance alignments in the asap dataset," *Trans. Int. Soc. Music. Inf. Retr.*, vol. 6, pp. 27–42, 2023. URL: https://api.semanticscholar.org/CorpusID:259361896.

[63] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, "Asap: A dataset of aligned scores and performances for piano transcription," in *International Society for Music Information Retrieval Conference*, 2020. URL: https://api.semanticscholar.org/CorpusID:221725841.

[64] J. Thickstun, J. Brennan, and H. Verma, "Rethinking evaluation methodology for audio-to-score alignment," *ArXiv*, vol. abs/2009.14374, 2020. URL: https://api.semanticscholar.org/CorpusID:222066842.

[65] P. A. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–I, 2001. URL: https://api.semanticscholar.org/CorpusID:2715202.

[66] G. Xia, Y. Wang, R. B. Dannenberg, and G. J. Gordon, "Spectral learning for expressive interactive ensemble music performance," in *International Society for Music Information Retrieval Conference*, 2015. URL: https://api.semanticscholar.org/CorpusID:1324539.