TISMIR

Henkel, F., et al. (2019). Score Following as a Multi-Modal Reinforcement Learning Problem. *Transactions of the International Society for Music Information Retrieval,* 2(1), pp. 67–81. DOI: https://doi.org/10.5334/tismir.31

RESEARCH

Score Following as a Multi-Modal Reinforcement Learning Problem

Florian Henkel*, Stefan Balke*, Matthias Dorfer* and Gerhard Widmer*,†

Score following is the process of tracking a musical performance (audio) in a corresponding symbolic representation (score). While methods using computer-readable score representations as input are able to achieve reliable tracking results, there is little research on score following based on raw score images. In this paper, we build on previous work that formulates the score following task as a multi-modal Markov Decision Process (MDP). Given this formal definition, one can address the problem of score following with state-of-the-art deep reinforcement learning (RL) algorithms. In particular, we design end-to-end multi-modal RL agents that simultaneously learn to listen to music recordings, read the scores from images of sheet music, and follow the music along in the sheet. Using algorithms such as synchronous Advantage Actor Critic (A2C) and Proximal Policy Optimization (PPO), we reproduce and further improve existing results. We also present first experiments indicating that this approach can be extended to track real piano recordings of human performances. These audio recordings are made openly available to the research community, along with precise note-level alignment ground truth.

Keywords: Reinforcement Learning; Score Following; Sheet Music

1. Introduction

Score following is a long-standing research problem in Music Information Retrieval (MIR). It lies at the heart of applications such as automatic page turning (Arzt et al., 2008), automatic accompaniment (Cont, 2010; Raphael, 2010) or the synchronization of visualizations in live concerts (Arzt et al., 2015; Prockup et al., 2013). Score following can be seen as an online variant of music synchronization where the task is to align a given music recording to its corresponding musical score (see Müller, 2015; Thomas et al., 2012, for overviews). However, in score following scenarios, the music recording is not known a priori and the systems need to react to the ongoing performance. Many traditional systems use online variants of dynamic time warping (DTW) (Dixon and Widmer, 2005; Arzt, 2016) or hidden Markov models (Orio et al., 2003; Cont, 2006; Schwarz et al., 2004; Nakamura et al., 2015). However, these approaches usually rely on a symbolic, computer-readable representation of the score, such as MusicXML or MIDI. This symbolic representation is created either manually (e.g., through the time-consuming process of (re-)setting the score in a music notation program), or automatically, via optical music recognition (OMR) (Hajič jr and Pecina, 2017; Byrd and Simonsen, 2015; Balke et al., 2015), which—depending of the quality of the scanned sheet music—may require additional manual checking and corrections. To bypass these additional steps, Dorfer et al. (2016) propose a multi-modal deep neural network that directly learns to match sheet music and audio in an end-to-end fashion. Given short excerpts of audio and the corresponding sheet music, the network learns to predict which location in the given sheet image best matches the current audio excerpt. In this setup, score following can be formulated as a multi-modal localization task.

Recently, Dorfer et al. (2018b) formulated the score following task as a Markov Decision Process (MDP), which enabled them to use state-of-the-art deep reinforcement learning (RL) algorithms to teach an agent to follow along an audio recording in images of scanned sheet music, as depicted in **Figure 1**. The task of the agent is to navigate through the score by adapting its reading speed in reaction to the currently playing performance. As ground truth for this learning task, we assume that we have a collection of piano pieces represented as aligned pairs of audio recordings and sheet music images. The preparation of such a collection, including the entire alignment process, is described in detail by Dorfer et al. (2018a). In general, this scenario constitutes an interesting research framework that addresses aspects of both the application of multi-modal learning and following in music, and advanced reinforcement learning.

* Johannes Kepler University Linz, AT

† Austrian Research Inst. for Artificial Intelligence, Vienna, AT

Corresponding author: Florian Henkel (florian.henkel@jku.at)

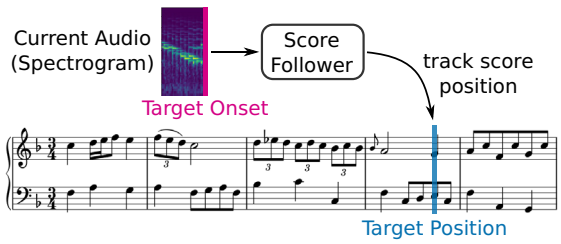

Figure 1: Sketch of score following in sheet music. Given the incoming audio, the score follower has to track the corresponding position in the score (image).

The specific contributions of the present work are as follows:

1. Based on the findings of Dorfer et al. (2018b), we extend the experiments with an additional policy gradient method, namely, Proximal Policy Optimization (PPO) (Schulman et al., 2017). Using PPO for our score-following scenario further improves the system's performance. This confirms one of the concluding hypotheses of Dorfer et al. (2018b), that improved learning algorithms directly translate into an improvement in our application scenario.
2. We provide extensive baseline experiments using optical music recognition and an online variant of DTW. The results indicate that our RL approach is a viable alternative to the OMR-DTW strategy, yielding competitive performance on the used datasets without additional preprocessing steps such as OMR.
3. All experiments so far were based on synthetic data, with audio synthesized directly from the score. We report on first experiments with recordings of 16 real piano performances. The recorded pieces belong to the test split of the Multi-modal Sheet Music Dataset (MSMD) (Dorfer et al., 2018a), which is also used in our other experiments. The results on this new dataset suggest that our agents are starting to generalize to the real-world scenario even though they were solely trained on synthetic data.
4. We make this set of piano recordings openly available to the research community, along with the ground-truth annotations (precise alignments of played notes to corresponding note heads in the sheet music).

In addition to quantitative experiments, we also take a look at the feature space learned by our agents, and what aspects of the state representation they tend to base their decisions on. To this end, we briefly present a t-SNE projection of a specific hidden network layer, and use a gradient-based attribution method to pinpoint what the agent "looks at" when making a decision. This allows for a sanity check of both model design and model behavior.

The remainder of the article is structured as follows. In Section 2, we start by defining the task of score following as a Markov Decision Process (MDP) and explaining its basic building blocks. Section 3 introduces the concept of Policy Gradient Methods and provides details on three learning algorithms we will use. Section 4 proceeds with a description of our experiments and presents results for the case of synthesized piano data. Section 5 then briefly looks at model interpretability, providing some glimpses into the learned representations and policies. In Section 6, we report on first experiments using real piano recordings instead of synthetic data. Finally, Section 7 summarizes our work and provides an outlook on future research directions.

2. Score Following as a Markov Decision Process

In this section, we formulate the task of score following as a Markov Decision Process (MDP), the mathematical foundation for reinforcement learning or, more generally, for the problem of sequential decision making. The notation in this paper closely follows the descriptions given in the book by Sutton and Barto (2018).[1]

Reinforcement learning can be seen as a computational approach to learning from interactions to achieve a certain predefined goal. **Figure 2** provides an overview of the components involved in the score following MDP. The score following agent (or learner) is the active component that interacts with its environment, which in our case is the score following task. The interaction takes place in a closed loop where the environment confronts the agent with a new situation (a state S_t) and the agent has to respond by making a decision, selecting one out of a predefined set of possible actions A_t. After each action taken the agent receives the next state S_{t+1} and a numerical reward signal R_{t+1} indicating how well it is doing in achieving the overall goal. Informally, the agent's goal in our case is to track a performance in the score as accurately and robustly as possible; this criterion will be formalized in terms of an appropriate reward signal in Section 2.3. By running the MDP interaction loop we end up with a sequence of states, actions, and rewards $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$, which is the kind of experience an RL agent is learning its behavior from. We will elaborate on different variants of the learning process in Section 3. The remainder of this section specifies all components of the score following MDP in detail. In practice, our MDP is implemented as an environment in OpenAI-Gym, an open source toolkit for developing and comparing reinforcement learning algorithms (Brockman et al., 2016).

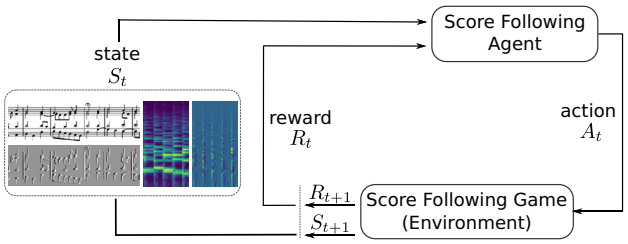

Figure 2: Sketch of the score following MDP. The agent receives the current state of the environment S_t and a scalar reward signal R_t for the action taken in the previous time step. Based on the current state it has to choose an action (e.g., decide whether to increase, keep or decrease its speed in the score) in order to maximize future reward by correctly following the performance in the score.

2.1. Score Following Markov States

Our agents need to operate on two different inputs at the same time, which together form the state S_t of the MDP: input modality one is a sliding window of the sheet image of the current piece, and modality two is an audio spectrogram excerpt of the most recently played music (~ 2 seconds). **Figure 3** shows an example of this input data for a piece by J. S. Bach. Given the audio excerpt as an input the agent's task is to navigate through the global score to constantly receive sheet windows from the environment that match the currently playing music. How this interaction with the score takes place is explained in the next subsection. The important part for now is to note that score following embodies dynamics which have to be captured by our state encoding, in order for the process to satisfy the Markov property. The Markov property means that a future state only depends on the current state, not on the past, i.e., $p(S_{t+1}|S_t, S_{t-1}, S_{t-2}, \ldots, S_0) = p(S_{t+1}|S_t)$. While there are ways to tackle problems where the Markov property is violated, it is desirable to formalize environments in such a way that the state transition process is Markovian (Sutton and Barto, 2018). Therefore, we extend the state representation by adding the one step differences (Δ) of both the score and the spectrogram. With the Δ-image of the score and the Δ-spectrogram, a state contains all the information needed by the agent to determine where and how fast it is moving along in the sheet image.

2.2. Agents, Actions, and Policies

The next item in the MDP (**Figure 2**) is the agent, which is the component interacting with the environment by taking actions as a response to states received. As already mentioned, we interpret score following as a multi-modal control problem where the agent decides how fast it needs to progress in the score. In more precise terms, the agent controls its score progression speed v_{pxl} in pixels per time step by selecting action $A_t \in \mathcal{A} := \{-\Delta v_{pxl}, 0, +\Delta v_{pxl}\}$ after receiving state S_t in each time step $t \in \mathbb{N}_0$. Actions $\pm\Delta v_{pxl}$ increase or decrease the speed by the constant Δv_{pxl} pixels per time step. Action $A_t = 0$ keeps it unchanged. To give an example: a pixel speed of $v_{pxl} = 14$ would shift the sliding sheet window 14 pixels forward (to the right) in the global unrolled score. Restricting the action space to three possible actions is a design choice and in theory one could use arbitrarily many actions or even a continuous action space. However, initial experiments showed that this is harder to learn for the agent and leads to an overall worse performance. Theoretically, our restricted set of actions might cause problems if a piece starts very fast, as the agent would not be able to speed up quickly enough; however, we did not find this to be a problem in our data.

Finally, we introduce the concept of a policy $\pi(a|s)$ to define an agent's behavior. π is a conditional probability distribution over actions conditioned on the current state. Given a state s, it computes an action selection probability $\pi(a|s)$ for each of the candidate actions $a \in \mathcal{A}$. The probabilities are then used for sampling one of the possible actions. In Section 3 we will parameterize policy $\pi(a|s;\boldsymbol{\theta})$ and explain how it can be learned by using deep neural networks as function approximators.

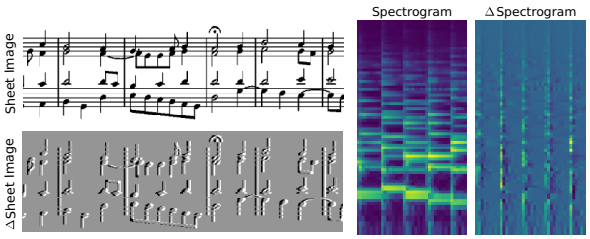

Figure 3: Markov state of the score following MDP: the current sheet sliding window and spectrogram excerpt. To capture the dynamics of the environment we also add the one step differences (Δ) w.r.t. the previous time step (state).

2.3. Goal Definition: Reward Signal and State Values

In order to learn a useful action selection policy, the agent needs feedback. This means that we need to define how to report back to the agent how well it does in accomplishing the task and, more importantly, what the task actually is.

The one component in an MDP that defines the overall goal is the reward signal $R_t \in \mathbb{R}$. It is provided by the environment in the form of a scalar, each time the agent performs an action. The sole objective of an RL agent is to maximize the cumulative reward over time. Note that achieving this objective requires foresight and planning, as actions leading to high instantaneous reward might lead to unfavorable situations in the future, and vice versa. To quantify this long-term success, RL introduces the return G, defined as the discounted cumulative future reward: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$. The discount rate $\gamma \in (0, 1]$ is a hyper-parameter assigning less weight to future rewards.

Figure 4 summarizes the reward computation in our score following MDP. Given annotated training data as mentioned in the introduction, the environment knows, for each onset time in the audio, the true target position x in the score. From this, and the current position $\hat{x}$ of the agent, we compute the current tracking error as $d_x = \hat{x} - x$, and define the reward signal $r(d_x)$ within a predefined tracking window $[x - b, x + b]$ around target position x as:

$$r(d_x) = 1.0 - \frac{|d_x|}{b}. \tag{1}$$

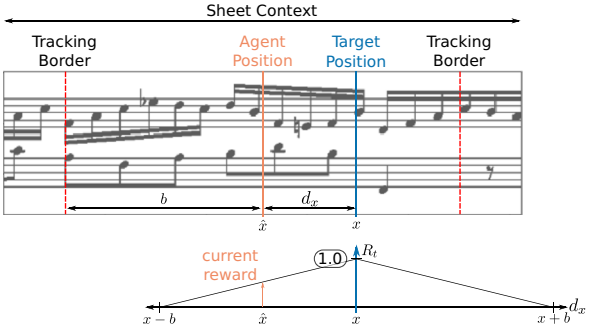

Figure 4: Reward definition in the score following MDP. The reward R_t (range [0, 1]) decays linearly with the agent's distance d_x from the current true score position x.

Thus, the reward per time step reaches its maximum of 1.0 when the agent's position is identical to the target position, and decays linearly towards 0.0 as the tracking error reaches the maximum permitted value b given by the window size. Whenever the absolute tracking error exceeds b (the agent drops out of the window), we reset the score following game (back to start of score, first audio frame). As an RL agent's sole objective is to maximize cumulative future reward G_t, it will learn to match the correct position in the score and not to lose its target by dropping out of the window. We define the target onset, corresponding to the target position in the score, as the rightmost frame in the spectrogram excerpt. This allows to run the agents on-line, introducing only the delay required to compute the most recent spectrogram frame. In practice, we linearly interpolate the score positions for spectrogram frames between two subsequent onsets in order to produce a continuous and stronger learning signal for training.

We will further define the State-Value Function as $v_\pi(s) := \mathbb{E}[G_t \mid S_t = s]$ which is the expected return given that we are in a certain state s and follow our policy π. Intuitively, this measures how good a certain state actually is: it is beneficial for the agent to be in a state with a high value as it will yield a high return in the long run. As with policy π, we use function approximation to predict the state value, denoted by $\hat{v}(s;\boldsymbol{w})$ with parameters $\boldsymbol{w}$. We will see in the next section how state-of-the-art RL algorithms use these value estimates to stabilize the variance-prone process of policy learning.

3. Learning To Follow

Given the formal definition of score following as an MDP we now describe how to address it with reinforcement learning. While there is a large variety of RL algorithms, we focus on policy gradient methods, in particular the class of actor-critic methods, due to their reported success in solving control problems (Duan et al., 2016). The learners utilized are REINFORCE with Baseline (Williams, 1992), Synchronous Advantage Actor Critic (A2C) (Mnih et al., 2016; Wu et al., 2017), and Proximal Policy Optimization (PPO) (Schulman et al., 2017), where the latter two are considered state-of-the-art approaches.

3.1. Policy and State-Value Approximation via DNNs

In Section 2, we introduced the policy π, determining the behavior of an agent, and value function $\hat{v}$, predicting how good a certain state s is with respect to cumulative future reward. Actor-critic methods make use of both concepts. The actor is represented by the policy π and is responsible for selecting the appropriate action in each state. The critic is represented by the value function $\hat{v}$ and helps the agent to judge how good the selected actions actually are. In the context of deep RL, both functions are approximated via Deep Neural Networks (DNNs), termed policy and value networks. In the following we denote the parameters of the policy and value network as $\boldsymbol{\theta}$ and $\boldsymbol{w}$, respectively.

Figure 5 shows a sketch of our architecture. Like Dorfer et al. (2016), we use a multi-modal convolutional neural network operating on both sheet music and audio at the same time. The input to the network is exactly the Markov state of the MDP introduced in Section 2.1. The left part of the network processes sheet images, the right part spectrogram excerpts (including Δ images). After low-level representation learning, the two modalities are merged by concatenation and further processed using dense layers. This architecture implies that policy and value networks share the parameters of the lower layers, which is a common choice in RL (Mnih et al., 2016). Finally, there are two output layers: the first represents our policy and predicts the action selection probability $\pi(a|s;\boldsymbol{\theta})$. It contains three output neurons (one for each possible action) converted into a valid probability distribution via soft-max activation. The second output layer consists of a single linear output neuron predicting the value $\hat{v}(s;\boldsymbol{w})$ of the current state. In Section 4 we list the exact architectures used for our experiments.

3.2. Learning a Policy via Policy Gradient

One of the first algorithms proposed for optimizing a policy was REINFORCE (Williams, 1992), a Monte Carlo algorithm that learns by generating entire episodes $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots$ of states, actions and rewards by following its current policy π while interacting with the environment. Given this sequence it then updates the parameters $\boldsymbol{\theta}$ of the policy network according to the following update rule, replaying the episode time step by time step:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\, G_t\, \nabla_{\boldsymbol{\theta}} \ln \pi\big(A_t \mid S_t; \boldsymbol{\theta}\big), \tag{2}$$

where α is the step size or learning rate and G_t is the true discounted cumulative future reward (the return) received from time step t onwards. Gradient $\nabla_{\boldsymbol{\theta}}$ is the direction in parameter space in which to go if we want to increase the

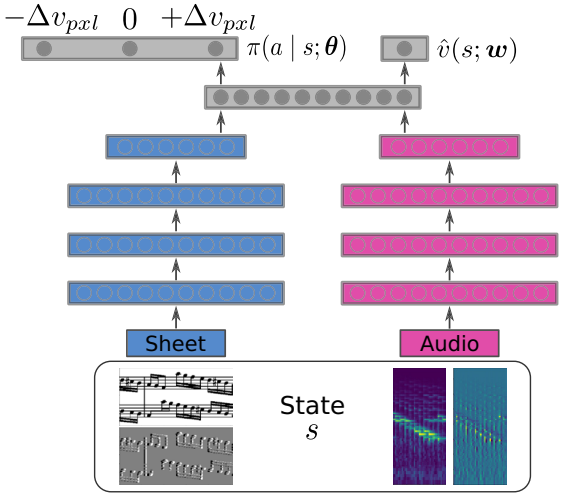

Figure 5: Multi-modal network architecture used for our score following agents. Given state s, the policy network predicts the action selection probability $\pi(a|s;\boldsymbol{\theta})$ for the allowed actions $a \in \{-\Delta v_{pxl}, 0, +\Delta v_{pxl}\}$. The value network, sharing parameters with the policy network, provides a state-value estimate $\hat{v}(s;\boldsymbol{w})$ for the current state.

selection probability of the respective action. This means whenever the agent did well (achieved a high return G_t), we take larger steps in parameter space towards selecting the responsible actions. By changing the parameters of the policy network, we of course also change our policy (behavior) and we will select beneficial actions more frequently in the future when confronted with similar states.

REINFORCE and policy optimization are known to have high variance in the gradient estimate (Greensmith et al., 2004). This results in slow learning and poor convergence properties. To address this problem, REINFORCE with Baseline ($\text{REINFORCE}_{\text{bl}}$) adapts the update rule of Equation (2) by subtracting the estimated state value $\hat{v}(s;\boldsymbol{w})$ (see Section 2.3) from the actual return G_t received:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \left(G_t - \hat{v}\left(S_t;\boldsymbol{w}\right)\right) \nabla_{\boldsymbol{\theta}} \ln \pi\left(A_t \mid S_t;\boldsymbol{\theta}\right). \tag{3}$$

This simple adaptation helps to reduce variance and improves convergence. The intuition behind subtracting $\hat{v}$ (the baseline) is that, as this term represents the expected or average return, we evaluate the actions we took in a certain state with respect to the average performance we expect in this exact state. This means that if we chose actions that were better than our average performance, we will increase the probability of taking them in the future, as the expression inside the brackets will be positive. If they were worse, the expression will be negative and we thus reduce their probabilities. The value network itself is learned by minimizing the squared difference between the actually received return G_t and the value estimate $\hat{v}(s;\boldsymbol{w})$ predicted by the network:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_{\boldsymbol{w}} \nabla_{\boldsymbol{w}} \left(G_t - \hat{v}\left(S_t;\boldsymbol{w}\right)\right)^2, \tag{4}$$

where $\alpha_{\boldsymbol{w}}$ is a separate learning rate for the value network. Note that as the policy and value network share some parameters, we can also jointly optimize them with a single learning rate. $\text{REINFORCE}_{\text{bl}}$ will be the first learning algorithm considered in our experiments.

Actor-critic methods are an extension of the baseline concept, allowing agents to learn in an online fashion while interacting with the environment. This avoids the need for creating entire episodes prior to learning and allows the agent to incrementally improve its policy step by step. In particular, our actor-critic agent will only look into the future for a fixed number of t_{max} time steps (in our case 15). This implies that we do not have the actual return G_t available for updating the value function. The solution is to bootstrap the value function (i.e., update the value estimate with estimated values), which is the core characteristic of actor-critic methods.

3.3. Advantage Actor Critic

In this work, we use the Advantage Actor Critic (A2C) algorithm, a synchronous version of the Asynchronous Advantage Actor Critic (A3C) (Mnih et al., 2016; Wang et al., 2016; Wu et al., 2017). One of the core aspects of this algorithm is to run multiple actors (in our case 8) in parallel on different instances of the same kind of environment, which should further help to stabilize training by decorrelating the samples used for updating. We will see in our experiments that this also holds for the score following task and the concept of parallelism further allows us to train the agents faster compared to algorithms like REINFORCE.

The algorithm itself can be seen as a multi-step actor-critic method, i.e., we take a certain number t_{max} of steps before we apply an update. Furthermore, the algorithm applies entropy regularization as introduced by Williams and Peng (1991): the entropy of the policy is added to the update rule, which should avoid early convergence to non-optimal policies as well as encourage exploration. The idea is to keep the entropy high and thus have more evenly distributed action selection probabilities. As a consequence, different actions will be chosen more frequently which in turn leads to more exploration. For readers unfamiliar with RL, this refers to the general trade-off between exploration and exploitation. Usually we try to learn a policy that is optimal in the sense of yielding the overall highest return. However, an agent that only executes those actions it currently thinks are the best (exploitation), might not discover more rewarding ones. In order to do so, the agent needs to try (explore) all actions to determine which are the best. This, however, contradicts the notion of optimality, as the agent will inevitably have to perform actions that are non-optimal in the long run. We found entropy regularization as a means of exploration to be crucial in our experiments, and the hyper-parameter controlling the influence of this regularization term requires careful tuning.

3.4. Proximal Policy Optimization (PPO)

The final algorithm we consider is Proximal Policy Optimization (Schulman et al., 2017). Similar to A2C, PPO is an actor-critic method; it employs the same approach of multiple parallel actors and multi-step updates. However, the objective and strategy for optimizing the policy are conceptually different.

One problem with regular policy gradient methods is sample efficiency, i.e., the number of interactions the agent has to perform within the environment until it is able to solve a given task. High variance in the gradient estimate during the learning process leads to a low sample efficiency and thus the goal is to reduce variance through various methods. For example, PPO uses generalized advantage estimation (GAE) (Schulman et al., 2015). GAE allows us to reduce variance by introducing bias (the well-known bias–variance trade-off (Bishop, 2006)). How much bias we introduce is controlled by an additional hyperparameter $\lambda \in [0, 1]$, where a value closer to zero will result in more bias. Besides GAE, PPO tries to improve sample efficiency further by performing multiple update steps, reusing samples created during interaction. However, in order to do so the objective used for optimizing the policy must be changed, as it would otherwise have a damaging effect on the policy itself (Schulman et al., 2017). Thus, PPO optimizes a clipped surrogate objective function allowing

to run multiple epochs of mini-batch updates on the policy. The idea behind this clipping objective is that an update will not drive the new policy too far away from the old one. The clipping itself is controlled by another hyperparameter $\varepsilon \in \mathbb{R}_{>0}$ that can significantly influence the results and therefore again requires proper tuning.

3.5. Distinction to Supervised Approaches

While RL offers appealing methods for solving the score following MDP, the question remains what advantages there are compared to a fully supervised approach. Considering the fact that algorithms like A2C and PPO have additional hyperparameters requiring careful tuning, and considering the long training times (for our MSMD data, this can be up to five days), it might seem reasonable to opt for supervised learning. This is theoretically possible, as we have an exact ground truth: a matching between time positions in the audio and the corresponding pixel positions in the score image. Thus we can derive an optimal tempo curve in terms of pixel speed as depicted in **Figure 6** and, consequently, an optimal sequence of tempo change actions for each of the training pieces. However, this optimal tempo sequence only offers a continuous solution (e.g., $A_t = 2.5$) and is not directly compatible with the discrete action space we defined in our MDP. As several RL algorithms such as PPO and Deep Deterministic Policy Gradient (Lillicrap et al., 2015) are applicable to continuous control problems, we could formulate the MDP in such a way. However, this not only makes the problem harder to learn for the algorithms, but also reintroduces the issue with ambiguous repetitive musical structures as described by Dorfer et al. (2016): there are situations where the current audio excerpts can match to multiple positions within the score excerpt, e.g., a repeatedly played note sequence. If the agent is allowed to freely adapt its position within the score, it is possible that it will jump between such ambiguous structures. The discrete action definition of our MDP alleviates this problem, as such jumps are not possible. In other words, the agent is constrained to read the score in small incremental pixel steps.

Taking a closer look at an example of such an optimal tempo curve (**Figure 6**), we observe another problem. The corresponding optimal actions we need as targets for training a supervised agent are zero most of the time. For the remaining steps we observe sparse spikes of varying amplitude. These sparse spikes are hard for a neural network to learn. In fact, we tried training a neural network with a similar structure as given in **Figure 5**, but ended up with a model that predicts values close to zero for all its inputs. Another drawback of the supervised approach is that during training, the agent would only see situations that correspond to the optimal trajectory. Thus, it would never encounter states where it has to recover itself in order to reach the desired target position. While this could be alleviated through sophisticated data augmentation, the RL framework offers a much more natural solution, as agents are inevitably confronted with imperfect situations, especially at the beginning of the training process.

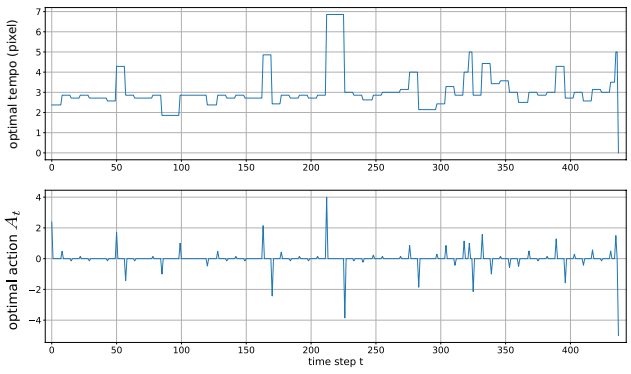

Figure 6: Optimal tempo curve and corresponding optimal actions A_t for a continuous agent (piece: J. S. Bach, BWV994). The A_t would be the target values for training an agent with supervised, feed-forward regression.

4. Experiments

In the following section, we reproduce the experiments reported by Dorfer et al. (2018b), and extend them with new results obtained with Proximal Policy Optimization (PPO), demonstrating additional improvement brought about by the new deep RL algorithm. Furthermore, we compare our RL approach to two baseline methods, one of which is based on the more "traditional" OMR+DTW pipeline already mentioned in the introduction. In Section 6, we complement this with first results on real human performances.

4.1. Experimental Setup

We use two different datasets in our experiments. The first one is a subset of the Nottingham Dataset, comprising 296 monophonic melodies of folk music, partitioned into 187 training, 63 validation and 46 test pieces (Boulanger-Lewandowski et al., 2012). The second one is the Multi-modal Sheet Music Dataset (MSMD) (Dorfer et al., 2018a). The original dataset consists of 479 classical pieces by various composers such as Bach, Mozart, and Beethoven. When visually exploring what our agents learn, we discovered alignment errors in 12 pieces (6 in the training and test split, respectively).[2] The cleaned version now consists of 354 training, 19 validation and 94 test pieces. In both cases the sheet music is typeset with Lilypond[3] and the audio is synthesized from MIDI using a piano sound font with a sample rate of 22.05 kHz. This automatic rendering process provides the precise audio-sheet music alignments required for training and evaluation. For audio processing we set the computation rate to 20 frames per second and compute log-frequency spectrograms. The FFT is computed with a window size of 2048 samples and post-processed with a logarithmic filterbank allowing only frequencies from 60 Hz to 6 kHz (78 frequency bins).

The spectrogram context visible to the agents is set to 40 frames (2 seconds of audio) and the sliding window sheet images of the unrolled score cover 40 × 150 and 160 × 512 pixels for Nottingham and MSMD, respectively. For MSMD we further downscale the score by a factor of two before presenting it to the network. The network architectures for Nottingham and MSMD are listed in **Table 1** and **Table 2**. We use exponential linear units (Clevert et al., 2016) for all but the two output layers. As optimizer we

Table 1: Network architecture used for the Nottingham dataset. Conv (3, stride-1)-16: 3 × 3 convolution, 16 feature maps and stride 1. No zero-padding is applied. We use ELU activation on all layers if not stated otherwise.

Audio (Spectrogram) 78 × 40	**Sheet-Image 40 × 150**
Conv (3, stride-2)-16	Conv (5, stride-(1, 2))-16
Conv (3, stride-2)-32	Conv (3, stride-2)-32
Conv (3, stride-2)-32	Conv (3, stride-2)-32
Conv (3, stride-1)-64	Conv (3, stride-(1,2))-64
Concatenation + Dense (256)	
Dense (256)	Dense (256)
Dense (3) – Softmax	Dense (1) – Linear

Table 2: Network architecture used for MSMD. DO: Dropout; Conv (3, stride-1)-16: 3 × 3 convolution, 16 feature maps and stride 1. No zero-padding is applied. We use ELU activation on all layers if not stated otherwise.

Audio (Spectrogram) 78 × 40	**Sheet-Image 80 × 256**
Conv (3, stride-1)-16	Conv (5, stride-(1, 2))-16
Conv (3, stride-1)-16	Conv (3, stride-1)-16
Conv (3, stride-2)-32	Conv (3, stride-2)-32
Conv (3, stride-1)-32 + DO (0.2)	Conv (3, stride-1)-32 + DO (0.2)
Conv (3, stride-2)-64	Conv (3, stride-2)-32
Conv (3, stride-2)-96	Conv (3, stride-2)-64 + DO (0.2)
Conv (1, stride-1)-96 + DO (0.2)	Conv (3, stride-2)-96
Dense (512)	Conv (1, stride-1)-96 + DO (0.2)
	Dense (512)
Concatenation + Dense (512)	
Dense (256) + DO (0.2)	Dense (256) + DO (0.2)
Dense (3) – Softmax	Dense (1) – Linear

use the Adam update rule (Kingma and Ba, 2015) with an initial learning rate of 10^{-4} and default parameters for the running average coefficients (0.9 and 0.999). We then train the models until there is no improvement in the number of tracked onsets on the validation set for 50 epochs and reduce the learning rate by a factor of 10 twice. The tempo change action Δv_{pxl} is 1.0 pixel per time step for both datasets. **Table 5** in the Appendix summarizes the hyperparameters used for training the agents.

Recall from Section 2.3 and **Figure 4** that from the agent's position $\hat{x}$ and the ground truth position x, we compute the tracking error d_x. We fix the size of the tracking window by setting b to be a third of the width of the score excerpt (b = 50 for the Nottingham dataset and b = 170 for MSMD). This error is the basis for our evaluation measures. However, in contrast to training, in the evaluation we only consider time steps where there is actually an onset present in the audio. While interpolating intermediate time steps is helpful for creating a stronger learning signal (Section 2.3), it is not musically meaningful. Specifically, we will report the evaluation statistics mean absolute tracking error $|\overline{d_x}|$ as well as its standard deviation $std(|d_x|)$ over all test pieces. These two measures quantify the accuracy of the score followers. To also measure their robustness we calculate the ratio $R_{\text{on}} \in [0, 1]$ of overall tracked onsets as well as the ratio of pieces $R_{\text{tue}} \in [0, 1]$ tracked from beginning entirely to the end:

$$R_{\text{on}} = \frac{\#\text{onsets tracked within window}}{\#\text{onsets}}, \tag{5}$$

$$R_{\text{tue}} = \frac{\#\text{pieces tracked until the end}}{\#\text{pieces}}. \tag{6}$$

An onset counts as tracked if the agent reached it without dropping out of the tracking window. If all onsets of a piece are tracked, i.e., the agent did not drop out of the tracking window, we say the piece is tracked until the end.

4.2. Baseline Approaches

In the following, we compare our RL-based agents to two different baselines. The first is the approach described by Dorfer et al. (2016), which models score following as a multi-modal localization task (denoted by MM-Loc in the following): the sheet snippets are split into discrete buckets, and a neural network is trained in a supervised fashion to predict the most probable bucket given the current audio excerpt. Note that this approach works on the raw image data, but it does not take into account any additional temporal context (e.g., contiguity of frames and buckets).

For the second baseline, we apply a variant of Online Dynamic Time Warping (ODTW) (Dixon, 2005). This approach does not work on the raw sheet image data and thus requires further preprocessing, namely, Optical Music Recognition (OMR), which converts the scanned sheet music into a computer-readable format such as MIDI or MusicXML. We consider two different settings for this baseline approach.

In the first setting (which we will call MIDI-ODTW), we assume that we have an ideal OMR system that can perfectly extract a score, in the form of a MIDI file, from the sheet music; to simulate this, we directly use the MIDI data contained in the MSMD dataset. Matching this perfect score to the synthetic performances is an easy task, since both—score and performance—are the same audio recordings at the sample level. We thus consider this method as our theoretical upper bound.

In the second setting (OMR-ODTW), we use a standard OMR system to extract a (possibly flawed) MIDI file from the rendered score image. For our experiments, we chose the open source tool Audiveris.[4] Audiveris's output is in the form of a MusicXML file, which we convert to MIDI (OMR-MIDI) using Musescore.[5] To align the extracted score with the performance, we synthesize the MIDI data using FluidSynth,[6] extract chroma features (feature rate = 20 Hz)

from the synthesized score and the performance audio, and align them using ODTW (Müller, 2015).

Evaluating MIDI-ODTW is trivial as we have a given ground truth matching (the warping path aligning audio and score is the main diagonal of the DTW's cost matrix). However, for OMR-ODTW, we face the problem that the note-wise annotations are no longer valid, since the OMR system may introduce errors to the extracted score. Thus, comparing this baseline to the proposed RL approach is difficult, as we require notehead alignments between score and the audio to compute our evaluation metrics. To avoid additional manual annotations of the note onset positions in the extracted OMR-MIDI, we evaluate this baseline by measuring the offset of the tracking position relative to a perfect alignment, disregarding local tempo deviations. In other words: the "perfect" alignment would correspond to the main diagonal of the cost matrix (which is no longer a square matrix due to the overall tempo/duration difference between OMR-MIDI and performance MIDI). The underlying assumption here is that no additional bars are inserted or removed in the OMR output, which we verify by a visual inspection of the extracted data. Given these warping paths, we can compute the measures introduced in Section 4.1, by projecting back onto the sheet images.

4.3. Experimental Results

Table 3 provides a summary of our experimental results. Our goal was, on the one hand, to reproduce the results by Dorfer et al. (2018b) and, on the other, to underpin their claim that improvements in the field of RL will eventually lead to improvements in the score following task.

As described above, we considered two datasets of different complexity: monophonic music in the Nottingham data and polyphonic music in the MSMD data. **Table 3** indicates that our adapted implementation and retrained A2C models deliver similar results for both datasets as reported by Dorfer et al. (2018b), with some improvement in alignment precision mainly due to variance in RL. However, the new PPO algorithm brings considerable additional improvement. This does not manifest itself on the Nottingham dataset, which is obviously too simple. On the polyphonic music provided in the MSMD dataset, however, the difference becomes evident: PPO outperforms A2C by 5 percentage points in terms of the number of pieces successfully tracked until the end (R_{tue}), and by about 3 points in terms of tracked onsets (R_{on}). Thus, PPO learns more robust tracking behavior, leading also to a slight improvement regarding accuracy.

Also, the results indicate that thanks to our generic formulation of the tracking task, advancements in RL research indeed directly translate into improvements in score following. This is particularly noticeable when we consider the performance of the "old" $\text{REINFORCE}_{\text{bl}}$ algorithm which, while learning a usable policy for the monophonic dataset, clearly drops behind on the polyphonic MSMD. The new variance reduction methods incorporated in the more advanced A2C and PPO algorithms exhibit their full power here.

While conducting these experiments we made a point of keeping the score-following MDP and the training process the same for both datasets. This is in contrast to Dorfer et al. (2018b), who used a different action space for the Nottingham dataset ($\Delta v_{pxl} = \pm 0.5$ compared to $\Delta v_{pxl} = \pm 1.0$). Here, we only adapt the underlying neural network architecture, making it less deep for the simpler Nottingham data. In this way, we want to provide additional evidence for the generality and robustness of our MDP formulation and the RL algorithms (especially A2C and PPO).

Comparing the RL agents to the supervised localization baseline (MM-Loc), we see that the baseline achieves a lower tracking error. However, it manages to track only 55% of the pieces to the end, compared to the 81% of PPO.

When we compare the results to the ODTW baselines, we observe two things. First, if we had a perfectly working

Table 3: Comparison of score following approaches. MIDI-ODTW considers a perfectly extracted score MIDI file and aligns it to a performance with ODTW. OMR-ODTW does the same, but uses a score MIDI file extracted by an OMR system. MM-Loc is obtained by using the method presented by Dorfer et al. (2016) with a temporal context of 4 and 2 seconds for Nottingham and MSMD, respectively. For MSMD, we use the models from the references and re-evaluate them on the cleaned data set. For A2C, PPO and $\text{REINFORCE}_{\text{bl}}$ we report the average over 10 evaluation runs. The mean absolute tracking error and its standard deviation are given in centimeters.

	Nottingham (monophonic, 46 test pieces)				MSMD (polyphonic, 94 test pieces)			
Method	$\mathbf{R_{tue}}$	$\mathbf{R_{on}}$	$\overline{\lvert \mathbf{d_x} \rvert}$	**std** ($\lvert \mathbf{d_x} \rvert$)	$\mathbf{R_{tue}}$	$\mathbf{R_{on}}$	$\overline{\lvert \mathbf{d_x} \rvert}$	**std** ($\lvert \mathbf{d_x} \rvert$)
MIDI-ODTW (upper bound)	1.00	1.00	0.00	0.00	1.00	1.00	0.00	0.00
OMR-ODTW	0.89	0.95	0.04	0.09	0.77	0.87	0.63	0.98
MM-Loc (Dorfer et al., 2018b)	0.65	0.83	0.08	0.28	0.55	0.60	0.29	1.07
A2C (Dorfer et al., 2018b)	0.96	0.99	0.08	0.12	0.76	0.77	0.69	0.81
$\text{REINFORCE}_{\text{bl}}$	0.97	0.99	0.06	0.09	0.59	0.70	1.06	1.07
A2C	0.96	0.99	0.07	0.09	0.75	0.77	0.68	0.82
PPO	0.99	0.99	0.06	0.09	0.81	0.80	0.65	0.81

OMR system (MIDI-ODTW), the task itself would become trivial. This method is a theoretical upper bound on the score-following performance and its main purpose is to verify our ODTW implementation. Second, and more interesting, we see that we do not (yet) have such a flawless OMR system. On the Nottingham dataset PPO surpasses OMR-ODTW on all measures except the average alignment error. While OMR-ODTW still outperforms the best RL method in terms of tracked onsets and average alignment error on MSMD, PPO manages to track more pieces and also has a lower standard deviation of the alignment errors. These results are promising and indicate that the RL approach is reasonable and even competitive to existing methods, with the additional advantage of directly working on raw sheet images, without the need of an additional OMR step.

We further note the real-time capabilities of our system. On average (estimated over 1000 trials), it takes approximately 2.04 ms for the agent to process a new incoming frame (corresponding to 50 ms of audio).[7] This measure is independent of the piece length, as it is primarily determined by the duration of a forward path through the underlying neural network.

5. Taking a Look Inside

As with many neural network-based machine learning models, our agent is something of a black box (Krause et al., 2016). While the experiments show that it is able to track pieces from beginning to end, we do not know why and how an agent's decision is formed. In the following section we try to gain some insight both into how the learned models organise the state space internally, and how the agents "look at" a state when making a decision.

5.1. A Look into Embedding Space: t-SNE

The network architecture given in **Figure 5** is structured in such a way that at first the two modalities (score and audio) are processed separately. After several convolutional layers we arrive at a lower-dimensional feature representation for each modality which is flattened and processed by a dense layer. Both are then concatenated to a single feature vector and passed through another dense layer representing the embedding of the two modalities. We now take the output of this 512 dimensional embedding layer and apply t-SNE (t-distributed stochastic neighborhood embedding) (van der Maaten and Hinton, 2008) to project the feature vector into a two dimensional space. The idea behind t-SNE is to project high-dimensional vectors into a lower-dimensional space in such a way that samples that are similar/close to each other are also close to each other in the lower-dimensional space. **Figure 7** provides a visualization of the embeddings for all states that our best agent (PPO) visited when evaluated on the MSMD test split.

Each point in the plot corresponds to an audio–score input tuple (state s) and is given a color according to the predicted value $\hat{v}(s;\boldsymbol{w})$. Remember that this value encodes the agent's belief of how good a state is, i.e., how much return it can expect if it is currently in this state. After visually exploring the data, we emphasize three clusters. The first cluster (C1) in the upper left corner turns out to comprise the beginnings of pieces. The value of those states is in general high, which is intuitive as the agent can expect the cumulative future reward of the entire pieces. Note that the majority of the remaining embeddings have a similar value. The reason is that all our agents use a discounting factor $\gamma < 1$ which limits the temporal horizon of the tracking process. This introduces an asymptotic upper bound on the maximum achievable value.

The second cluster (C2) on the right end corresponds to the piece endings. Here we observe a low value as the agent has learned that it cannot accumulate more reward in such states due to the fact that the piece ends. The third cluster (C3) is less distinct. It contains mainly states with a clef somewhere around the middle of the score excerpt. (These are the result of our way of "flattening" our scores by concatenating single staff systems into one long unrolled score.) We observe mixed values that

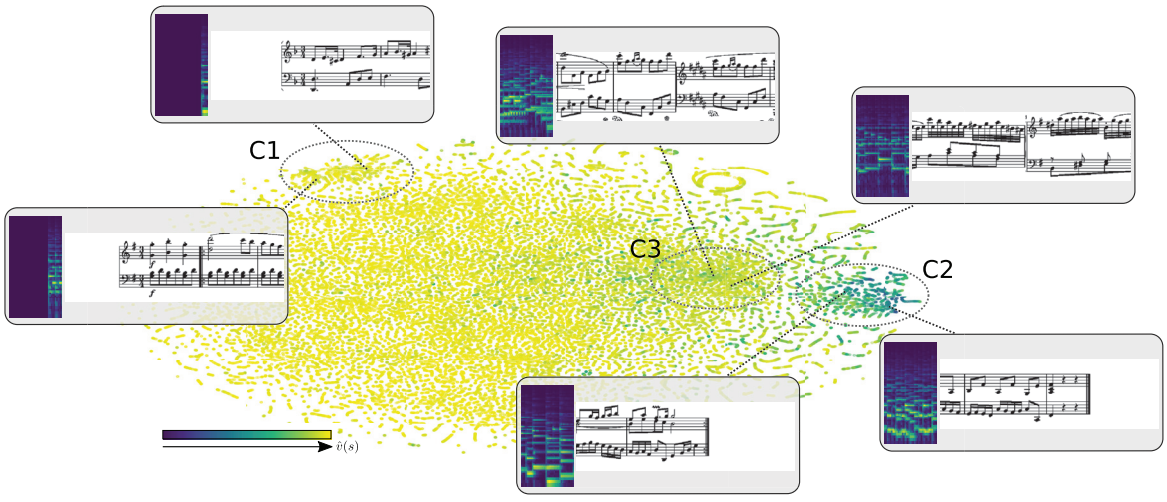

Figure 7: Two-dimensional t-SNE projection of the 512-dimensional embeddings taken from the network's concatenation layer (see Figure 5). Each point in the scatter plot corresponds to an audio–score input tuple. The color encodes the predicted value $\hat{v}(s;\boldsymbol{w})$. (Figure inspired by Mnih et al. (2015).)

lie in the middle of the value spectrum. A reason for this might be that these situations are hard for the agent to track accurately, because it has to rapidly adapt its reading speed in order to perform a "jump" over the clef region. This can be tricky, and it can easily happen that the agent loses its target shortly before or after such a jump. Thus, the agent assigns a medium value to these states.

5.2. A Look into the Policy: Integrated Gradients

A second relevant question is: what exactly causes the agent to choose a particular action? Different approaches to answering this have recently been explored in the deep learning community (Baehrens et al., 2010; Shrikumar et al., 2017; Sundararajan et al., 2017). One of these is called integrated gradients (Sundararajan et al., 2017). The idea is to explain an agent's decision by finding out which parts of the input were influential in the prediction of a certain action. This is done by accumulating the gradients of the prediction with respect to multiple scaled variations of the input, which the authors refer to as the path integral. The gradient with respect to the input points is the direction that maximizes the agent's decision, i.e., the probability of choosing the action that currently has the highest probability. Thus, a high gradient for a certain input feature suggests a high impact on the decision.

In the following, we briefly look at two examples from our score following scenario.[8] **Figure 8a** shows a situation where the agent is behind the true target position. This results in a high predicted probability for increasing the current pixel speed. In the contrary situation in **Figure 8b**, we observe the highest probability for decreasing the pixel speed. This makes intuitive sense, of course: when we are ahead of the target, the agent should probably slow down; when we are behind, it should speed up. **Figure 9** shows the corresponding integrated gradients as a salience map overlaid on top of the state representation, for the situation depicted in **Figure 8b**. In the upper left part, the agent's focus in the plain score is shown. We see that some of the note heads around the center (the current tracking position) are highlighted, which means that they have a strong impact on the agent's decision. If we further consider the spectrogram, it seems that the agent also

Figure 8: Two examples of policy outputs. **(a)** Agent is behind the target, resulting in a high probability for increasing the pixel speed ($\pi(+\Delta v_{pxl} \mid s;\boldsymbol{\theta}) = 0.795$). **(b)** Agent is ahead of the target, suggesting a reduction of pixel speed ($\pi(-\Delta v_{pxl} \mid s;\boldsymbol{\theta}) = 0.903$).

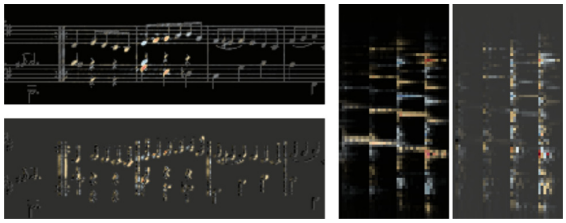

Figure 9: Visualization of the agent's focus on different parts of the input state for the situation shown in Figure 8b. The salience map was created via integrated gradients (Sundararajan et al., 2017), a technique to identify the most relevant input features for the agent's decision—in this case, for decreasing its pixel speed.

puts emphasis on the harmonics. Note also that some of the highlighted note heads match to the corresponding highlighted parts in the spectrogram.

A look at the delta score image (**Figure 9**, bottom left) reveals that the influential input parts are less sharply centered around the current tracking position than in the score's salience map. It seems that the agent generally attends to the pixel deltas, which encode the reading speed, regardless of the current musical content. Recalling the definition of the Markov state (see Section 2.1), this is reassuring to see, as this part of the input state was intentionally designed for exactly this purpose.

Regarding the spectrogram delta, we are undecided whether it has beneficial effects on policy learning. As the spectrogram is provided by the environment (one new frame per time step), there is no additional information captured in the delta spectrogram. Nevertheless, it may be beneficial as an additional input feature that highlights note onsets in the audio signal. Preliminary experiments indicate that we might be able to remove this additional input without losing tracking accuracy.

6. Investigations on Real Performances

So far, all experiments were based on synthesized piano music. However, in real performances, the agent is confronted with additional challenges, including local and global tempo deviations, playing errors, or varying acoustic conditions (Widmer, 2017). In order to evaluate these challenges in a controlled way, we asked two pianists to perform 16 pieces (split between them) from the MSMD test dataset. For these experiments, we only selected pieces from the test set where the agent exhibited acceptable performance on the synthesized data. During the recording session, the pianists were asked to perform the pieces without going to extremes. Still, the performances include local tempo deviations, as well as additional ornaments like trills (especially in the Bach and Mozart pieces; please consult **Table 6** in the Appendix for an overview of the recorded pieces).

The actual recordings took place at our institute in a regular office environment (room dimensions c. 5 × 6 meters). The instrument was a Yamaha AvantGrand N2 hybrid piano (A4 = 440 Hz). From the piano, we simultaneously recorded the MIDI signal (performance MIDI), the direct stereo output (dry), and the acoustic room signal by placing an omni-directional microphone about 2 meters away from the piano. The different signals

can be related to different levels of difficulty, e.g., the room signal is harder to track than the dry signal since additional room reflections influenced the recording.

To establish a ground truth, we used the performance MIDI files, the corresponding score MIDI file, and a rendered version of the score, where the latter two were taken from the MSMD dataset (i.e., the score images were generated with Lilypond). Using the alignment technique described by Dorfer et al. (2018a), we established the needed connection between note head positions in the scores and the corresponding notes in the performance MIDI file and audio recordings, respectively. Note that this alignment is automatically generated and not perfect for all of the notes.

To compare the performance of our RL agents in the context of real performances, we conduct four experiments with increasing level of difficulty. First, the algorithms are evaluated on the original synthesized MIDI score, which provides an upper performance bound: we do not expect the agents to do better on the set of real performances. Second, we synthesize the MIDI data we got from the real recordings with the same piano synthesizer used during training. This experiment is meant to tell us how the agents cope with performance variations that are not necessarily encoded in the score, but keeping the audio conditions unchanged. For the third and fourth experiments, we use the audio from the direct output of the piano and the room microphone, respectively, instead of synthesizing it from MIDI. This gives us insight into how the agents are able to generalize to real world audio. These four experiments comprise different challenges, where intuitively the first is the easiest and the last one should be the hardest due to noisy recording conditions. We compare our agents to the same baseline approaches as in Section 4. The ground truth for the ODTW baselines is derived as described in Section 4.2, by using the given automatic alignment.

The results of these experiments are summarized in **Table 4**. In general, we observe that real performances introduce a larger mean error and standard deviation. As expected, we also see a performance decrease with increasing difficulty: best results are achieved on the original synthesized MIDI, followed by the synthesized performance MIDI and the direct out recording (with the exception of $\text{REINFORCE}_{\text{bl}}$). For the room recordings we observe the weakest performance.

$\text{REINFORCE}_{\text{bl}}$ is again outperformed by both A2C and PPO, but contrary to the results on synthetic data we now observe that the PPO agent performs worse than the A2C agent on the real performances. It might be the case that PPO overfitted to the training conditions and is thus not able to deal with performance variations as well as A2C. However, the problem of overfitting in RL is difficult to address and the object of ongoing research efforts (e.g., Cobbe et al., 2018). Thus, further experiments with a larger test set are necessary to conclude if this is really the case.

Comparing the RL agents to MM-Loc shows that the supervised baseline does not generalize as well and has likely overfitted to the training conditions. The performance of the ODTW baselines is at a similar level over the different experimental settings; however, we see a higher performance deterioration for the OMR baseline (OMR-ODTW) compared to the results in Section 4.3, where score and performance are created from the same MIDI file. As these methods seem to be more robust against different recording conditions (most likely due to the chroma features used to represent the audio), they still exceed the machine learning based approaches in almost all cases. For future work it will be necessary to improve the generalization capabilities of the proposed approach by making it more robust to different acoustic scenarios, e.g., through data augmentation.

Table 4: Comparison of score following approaches on real performances. To get a more robust estimate of the performance of the RL agents ($\text{REINFORCE}_{\text{bl}}$, A2C and PPO), we report the average over 50 evaluation runs. MM-Loc is the supervised baseline presented by Dorfer et al. (2016). MIDI-ODTW and OMR-ODTW are the ODTW baselines described in Section 4.2. The mean absolute tracking error and its standard deviation are given in centimeters.

Method	R_{tue}	R_{on}	$\overline{\lvert d_x \rvert}$	$\text{std}(\lvert d_x \rvert)$
				Original MIDI Synthesized (Score = Performance)
MIDI-ODTW	1.00	1.00	0.00	0.01
OMR-ODTW	0.62	0.80	0.85	1.12
MM-Loc	0.44	0.45	0.38	1.14
$\text{REINFORCE}_{\text{bl}}$	0.56	0.59	1.15	1.14
A2C	0.70	0.63	0.65	0.82
PPO	0.74	0.68	0.7	0.87
				Performance MIDI Synthesized
MIDI-ODTW	0.81	0.94	0.50	0.76
OMR-ODTW	0.50	0.72	0.90	1.08
MM-Loc	0.25	0.51	0.36	0.99
$\text{REINFORCE}_{\text{bl}}$	0.14	0.31	1.80	1.48
A2C	0.58	0.51	0.94	0.94
PPO	0.56	0.50	0.94	1.01
				Direct Out
MIDI-ODTW	0.88	0.92	0.59	0.79
OMR-ODTW	0.50	0.67	0.93	1.15
MM-Loc	0.19	0.32	0.55	1.42
$\text{REINFORCE}_{\text{bl}}$	0.33	0.43	1.42	1.29
A2C	0.49	0.55	0.97	1.06
PPO	0.51	0.53	1.01	1.11
				Room Recording
MIDI-ODTW	0.81	0.93	0.64	0.84
OMR-ODTW	0.50	0.65	0.93	1.09
MM-Loc	0.00	0.19	0.68	1.58
$\text{REINFORCE}_{\text{bl}}$	0.08	0.37	1.52	1.34
A2C	0.38	0.50	1.11	1.12
PPO	0.30	0.43	1.26	1.24

7. Conclusion and Future Work

In this paper, we investigated the potential of deep RL for the task of online score following on raw sheet images. Using a more advanced learning algorithm than the one used by Dorfer et al. (2018b), we were able not only to reproduce the reported results but also to improve the tracking performance. Given that RL is currently one of the most actively researched areas in machine learning, we expect further advances that we think will directly transfer to score following.

Furthermore, we conducted first experiments involving real performances. While the initial results are promising, there is still a lot of room for improvement. Also, in contrast to most state-of-the-art methods for general music tracking, our method is currently restricted to piano music.

The RL framework as such can be adapted to a variety of different alignment scenarios, given appropriate data. In particular, the input modalities can be exchanged to handle audio–MIDI, audio–lyrics, or MIDI–MIDI alignment scenarios. The latter is of interest for piano accompaniment systems, where actions of the agent could involve triggering of events, e.g., playing an accompaniment in sync with a live performer.

Moreover, we are eager to see deep RL-based approaches in other MIR-related tasks, such as automatic music transcription. This is challenging, because coping with high-dimensional action spaces (e.g., in theory 2^{88} for piano music transcription) is still an open problem in RL research. Still, we think this is an attractive line of research to follow, because of the generality and conceptual simplicity of the reinforcement learning scenario: in principle, we only need to find an appropriate formulation of a task (including Markov state representation and, crucially, a way of generating a not-too-sparse reward signal), and we will immediately benefit from the power of RL with all the exciting developments currently going on in this research area.

Appendix

In Table 5 we provide a summary of all the hyperparameters used in the training process and for the RL algorithms. In Table 6 an overview of the pieces recorded for Section 6 is given.

Table 5: Hyperparameter overview.

Hyperparameter	Value
Adam learning rate	10^{-4}
Adam decay rates (β_1, β_2)	(0.9, 0.999)
Patience	50
Learning rate multiplier	0.1
Refinements	2
Time horizon t_{max}	15
Number of actors	8
Entropy regularization	0.05
Discount factor γ	0.9
GAE parameter λ	0.95
PPO clipping parameter ε	0.2
PPO epochs	1
PPO batch size	120

Table 6: Overview of the pieces from the MSMD dataset that were recorded as real performances. The pieces are played without repetitions.

Composer	Piece name	Dur. (sec.)
Bach, Johann Sebastian	Polonaise in F major, BWV Anh. 117a	47.32
Bach, Johann Sebastian	Sinfonia in G minor, BWV 797	99.69
Bach, Johann Sebastian	French Suite No. 6 in E major, Menuet, BWV 817	37.21
Bach, Johann Sebastian	Partita in E minor, Allemande, BWV 830-2	86.73
Bach, Johann Sebastian	Prelude in C major, BWV 924a	40.43
Bach, Johann Sebastian	Minuet in F major, BWV Anh. 113	40.49
Bach, Johann Sebastian	Minuet in G major, BWV Anh. 116	51.56
Bach, Johann Sebastian	Minuet in A minor, BWV Anh. 120	31.32
Chopin, Frédéric François	Nocturne in B♭ minor, Op. 9, No. 1	328.92
Mozart, Wolfgang Amadeus	Piano Sonata No. 11 in A major, 1st Movt, Variation 1, KV331	56.33
Mussorgsky, Modest Petrovich	Pictures at an Exhibition, Promenade III	27.17
Schumann, Robert	Album für die Jugend, Op. 68, 1. Melodie	45.50
Schumann, Robert	Album für die Jugend, Op. 68, 6. Armes Waisenkind	73.52
Schumann, Robert	Album für die Jugend, Op. 68, 8. Wilder Reiter	24.88
Schumann, Robert	Album für die Jugend, Op. 68, 16. Erster Verlust	55.83
Schumann, Robert	Album für die Jugend, Op. 68, 26. Untitled	74.40

Reproducibility

The data and code for reproducing our results, along with detailed instructions and further examples, are available online: https://github.com/CPJKU/score_following_game and on the accompanying website http://www.cp.jku.at/resources/2019_RLScoFo_TISMIR.

Notes

[1] As in Sutton and Barto (2018), we denote random variables with capital letters such as state S_t and instances with small letters such as *s*.

[2] The errors are caused by inconsistencies in the way "Da capo" is encoded in Lilypond.

[3] http://lilypond.org/.

[4] https://github.com/Audiveris/audiveris.

[5] https://musescore.org/.

[6] http://www.fluidsynth.org/.

[7] Tested on a system with a consumer GPU (NVIDIA GEFORCE GTX 1080), 16GB RAM and an Intel i7-7700 CPU.

[8] More examples and video renditions can be found on the paper's accompanying website http://www.cp.jku.at/resources/2019_RLScoFo_TISMIR.

Acknowledgements

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement number 670035, project "Con Espressione"). The authors would like to thank Jan Hajič jr and Carlos Eduardo Cancino Chacón—both wonderful colleagues and fabulous piano players—for helping to record the piano performances used in Section 6. Many thanks to the anonymous reviewers and the editors for very helpful (and also a bit challenging) comments and suggestions which helped to improve this manuscript.

Competing Interests

The authors have no competing interests to declare.

References

Arzt, A. (2016). Flexible and Robust Music Tracking. PhD thesis, Johannes Kepler University Linz.

Arzt, A., Frostel, H., Gadermaier, T., Gasser, M., Grachten, M., & **Widmer, G.** (2015). Artificial Intelligence in the Concertgebouw. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 2424–2430). Buenos Aires, Argentina.

Arzt, A., Widmer, G., & **Dixon, S.** (2008). Automatic Page Turning for Musicians via Real-Time Machine Listening. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)* (pp. 241–245). Patras, Greece.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & **Müller, K.-R.** (2010). How to Explain Individual Classification Decisions. *Journal of Machine Learning Research, 11*, 1803–1831.

Balke, S., Achankunju, S. P., & **Müller, M.** (2015). Matching Musical Themes Based on Noisy OCR and OMR Input. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 703–707). Brisbane, Australia. DOI: https://doi.org/10.1109/ICASSP.2015.7178060

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Boulanger-Lewandowski, N., Bengio, Y., & **Vincent, P.** (2012). Modeling Temporal Dependencies in High-dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In *Proceedings of the 29th International Conference on Machine Learning (ICML).* Edinburgh, UK. DOI: https://doi.org/10.1109/ICASSP.2013.6638244

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & **Zaremba, W.** (2016). OpenAI Gym. arXiv preprint arXiv:1606.01540.

Byrd, D., & **Simonsen, J. G.** (2015). Towards a Standard Testbed for Optical Music Recognition: Definitions, Metrics, and Page Images. *Journal of New Music Research, 44*(3), 169–195. DOI: https://doi.org/10.1080/09298215.2015.1045424

Clevert, D., Unterthiner, T., & **Hochreiter, S.** (2016). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *Proceedings of the International Conference on Learning Representations (ICLR)* (arXiv:1511.07289).

Cobbe, K., Klimov, O., Hesse, C., Kim, T., & **Schulman, J.** (2018). Quantifying Generalization in Reinforcement Learning. arXiv preprint arXiv:1812.02341.

Cont, A. (2006). Realtime Audio to Score Alignment for Polyphonic Music Instruments using Sparse Non-Negative Constraints and Hierarchical HMMs. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (vol. 5, pp. 245–248). Toulouse, France. DOI: https://doi.org/10.1109/ICASSP.2006.1661258

Cont, A. (2010). A Coupled Duration-Focused Architecture for Real-Time Music-to-Score Alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(6), 974–987. DOI: https://doi.org/10.1109/TPAMI.2009.106

Dixon, S. (2005). An On-Line Time Warping Algorithm for Tracking Musical Performances. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 1727–1728). Edinburgh, UK.

Dixon, S., & **Widmer, G.** (2005). MATCH: A music alignment tool chest. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)* (pp. 492–497). London, UK.

Dorfer, M., Arzt, A., & **Widmer, G.** (2016). Towards Score Following in Sheet Music Images. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)* (pp. 789–795). New York, USA.

Dorfer, M., Hajič, J., Jr., Arzt, A., Frostel, H., & **Widmer, G.** (2018a). Learning Audio–Sheet Music Correspondences for Cross-Modal Retrieval and Piece Identification. *Transactions of the International Society for Music Information Retrieval, 1*(1), 22–33. DOI: https://doi.org/10.5334/timsir.12

Dorfer, M., Henkel, F., & **Widmer, G.** (2018b). Learning to Listen, Read, and Follow: Score Following as a Reinforcement Learning Game. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)* (pp. 784–791). Paris, France.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., & **Abbeel, P.** (2016). Benchmarking Deep Reinforcement Learning for Continuous Control. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)* (pp. 1329–1338). New York City, United States.

Greensmith, E., Bartlett, P. L., & **Baxter, J.** (2004). Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *Journal of Machine Learning Research, 5*, 1471–1530.

Hajič, J., Jr. and **Pecina, P.** (2017). The MUSCIMA++ Dataset for Handwritten Optical Music Recognition. In *14th International Conference on Document Analysis and Recognition (ICDAR)* (pp. 39–46). New York, United States. DOI: https://doi.org/10.1109/ICDAR.2017.16

Kingma, D., & **Ba, J.** (2015). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)* (arXiv:1412.6980).

Krause, J., Perer, A., & **Ng, K.** (2016). Interacting with Predictions: Visual Inspection of Black-box Machine Learning Models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 5686–5697). ACM. DOI: https://doi.org/10.1145/2858036.2858529

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & **Wierstra, D.** (2015). Continuous Control with Deep Reinforcement Learning. arXiv preprint arXiv:1509.02971.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & **Kavukcuoglu, K.** (2016). Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)* (pp. 1928–1937). New York City, United States.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & **Hassabis, D.** (2015). Human-level Control Through Deep Reinforcement Learning. *Nature, 518*, 529–533. DOI: https://doi.org/10.1038/nature14236

Müller, M. (2015). Fundamentals of Music Processing. Springer Verlag. DOI: https://doi.org/10.1007/978-3-319-21945-5

Nakamura, E., Cuvillier, P., Cont, A., Ono, N., & **Sagayama, S.** (2015). Autoregressive Hidden Semi-Markov Model of Symbolic Music for Score Following. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)* (pp. 392–398). Málaga, Spain.

Orio, N., Lemouton, S., & **Schwarz, D.** (2003). Score Following: State of the Art and New Developments. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)* (pp. 36–41). Montreal, Canada.

Prockup, M., Grunberg, D., Hrybyk, A., & **Kim, Y. E.** (2013). Orchestral Performance Companion: Using Real-Time Audio to Score Alignment. *IEEE Multimedia, 20*(2), 52–60. DOI: https://doi.org/10.1109/MMUL.2013.26

Raphael, C. (2010). Music Plus One and Machine Learning. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 21–28).

Schulman, J., Moritz, P., Levine, S., Jordan, M., & **Abbeel, P.** (2015). High-dimensional Continuous Control Using Generalized Advantage Estimation. arXiv preprint arXiv:1506.02438.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & **Klimov, O.** (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.

Schwarz, D., Orio, N., & **Schnell, N.** (2004). Robust Polyphonic MIDI Score Following with Hidden Markov Models. In *International Computer Music Conference (ICMC)*. Miami, Florida, USA.

Shrikumar, A., Greenside, P., & **Kundaje, A.** (2017). Learning Important Features through Propagating Activation Differences. arXiv preprint arXiv:1704.02685.

Sundararajan, M., Taly, A., & **Yan, Q.** (2017). Axiomatic Attribution for Deep Networks. arXiv preprint arXiv:1703.01365.

Sutton, R. S., & **Barto, A. G.** (2018). Reinforcement Learning. MIT Press, 2nd edition.

Thomas, V., Fremerey, C., Müller, M., & **Clausen, M.** (2012). Linking Sheet Music and Audio – Challenges and New Approaches. In M. Müller, M. Goto, & M. Schedl (Eds.), *Multimodal Music Processing, volume 3 of Dagstuhl Follow-Ups* (pp. 1–22). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

van der Maaten, L., & **Hinton, G.** (2008). Visualizing Data Using t-SNE. *Journal of Machine Learning Research, 9*, 2579–2605.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., & **Botvinick, M.** (2016). Learning to Reinforcement Learn. arXiv preprint arXiv:1611.05763.

Widmer, G. (2017). Getting Closer to the Essence of Music: The Con Espressione Manifesto. *ACM Transactions on Intelligent Systems and Technology (TIST), 8*(2), 19. DOI: https://doi.org/10.1145/2899004

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning, 8*, 229–256. DOI: https://doi.org/10.1007/BF00992696

Williams, R. J., & **Peng, J.** (1991). Function Optimization Using Connectionist Reinforcement Learning Algorithms. *Connection Science, 3*(3), 241–268. DOI: https://doi.org/10.1080/09540099108946587

Wu, Y., Mansimov, E., Liao, S., Grosse, R. B., & **Ba, J.** (2017). Scalable Trust-region Method for Deep Reinforcement Learning Using Kronecker-factored Approximation. *CoRR*, abs/1708.05144.

How to cite this article: Henkel, F., Balke, S., Dorfer, M., & Widmer, G. (2019). Score Following as a Multi-Modal Reinforcement Learning Problem. *Transactions of the International Society for Music Information Retrieval,* 2(1), pp. 67–81. DOI: https://doi.org/10.5334/tismir.31

Submitted: 01 February 2019 **Accepted:** 12 September 2019 **Published:** 20 November 2019

]u[*Transactions of the International Society for Music Information Retrieval* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS