Piano-playing Tasks with Deep Reinforcement Learning and Recurrent Neural Network

Thanachart Satianjarukarn
63340500021
Institute of Field Robotics
King Mongkut's University of Technology Thonburi
Bangkok, Thailand
thanachart.st@mail.kmutt.ac.th

Worranittha Larpkiattaworn
63340500057
Institute of Field Robotics
King Mongkut's University of Technology Thonburi
Bangkok, Thailand
worranittha.larp@mail.kmutt.ac.th

***Abstract*—High-dimensional control, particularly for piano playing with dexterous hands, still remains a challenge in robotics. Reinforcement learning (RL) methods have been shown to address this challenge in recent years. However, the existing method did not leverage sequential patterns in musical pieces which could enhance an agent's learning ability. This study introduces an RL approach utilizing a recurrent network architecture, RNN-MLP network, to leverage the patterns in the musical pieces for piano-playing tasks with anthropomorphic hands. Our research demonstrates that the agent with RNN-MLP network outperforms the traditional MLP network, showing the improved learning ability with higher episode return and faster convergence. This study also reveals that too high discount factor has a negative effect, leading to instability and drop in the agent's performance. In addition, it concludes with the performance of an agent with RNN-MLP network in handling different characteristics of musical pieces.**

***Index Terms*—High-dimensional Control, Deep Reinforcement Learning, Recurrent Neural Network**

I. Introduction

High-dimensional control is one of the grand challenges in robotics. Several researches have been conducted to facilitate this task that range from a traditional control such as model predictive control (MPC) to reinforcement learning (RL) approach.

RoboPianist [12] suggests the recent solution for high-dimensional control in piano-playing tasks with human-like dexterous hands. It shows the potential of RL approach, resulting in improved performance of an agent for playing several musical pieces. However, this method does not leverage the patterns in musical pieces which can be utilized to improve the learning ability of an agent.

This study presents the process for taking advantage of the musical patterns by using the combination of the existing RL approach and a recurrent network, RNN-MLP network. This network demonstrates the ability to handle and leverage both sequential and static data from the musical pieces, showing the improved learning ability of an agent. In addition, our study also explores the effect of the discount factor, highlighting that too high discount factors can lead to instability in the learning process and drop an agent's performance.

II. Related work

Several piano-playing robots have been developed, showing the designs of hardware and controllers. Recently, the piano-playing task was formulated as the reinforcement learning (RL) problem for a single hand [10]. They designed an end-to-end RL method and demonstrated how an agent can learn from the musical pieces and play the piano with a dexterous hand on a simulated piano. The results showed the effectiveness of the RL approach which allows the agent to deal with piano-playing tasks and correctly play correct notes, velocity and fingering. However, due to the use of a single robot hand, There is still a gap compared to human capabilities.

Most recently, RoboPianist [12] has been developed to address this gap by dealing with high-dimensional control to create more realistic or human-like dexterity of robot hands. They applied the RL approach in the piano-playing task with two anthropomorphic hands, enabling the agent to learn piano pieces which the traditional methods still struggle with. However, their critic and policy networks are designed by only multi-layer perceptrons even the musical pieces have some patterns which can be utilized.

Recurrent Neural Network (RNN) is a type of neural network or a connectionist model which can be used for pattern recognition [6]. Since RNN has a memory for storing the information, it can handle sequential data while learning relationships or recognizing the patterns within the musical pieces.

In this study, we focus on improving the performance of an agent in the piano-playing tasks by leveraging the musical patterns and a combination of the RL approach with RNN.

III. Methodology

A. Dataset

In this study, due to the limitation of time, we select only one popular piece, named Für Elise by Beethoven, from the PIG dataset [7] which is a publicly available data containing 150 classical piano pieces with fingering annotations for using in our experiment. This song consists of 24 bars and 208 notes which are in length of 20 seconds. We use the Musical Instrument Digital Interface (MIDI) standard to represent this musical piece and convert it into a time-indexed note trajectory where each note is encoded to a one-hot vector with length 88 before feeding to the agent.

B. Simulation Details

In this work, we utilize the simulated piano-playing environment from RoboPianist [12] which uses the MuJoCo [8] physics engine. The piano model is a full-size digital keyboard with 52 white keys and 36 black keys which is modeled as a joint with a linear spring [5]. Each key is active when its joint position is within 0.5 degrees of its maximum range. This environment also has a mechanism of a sustain pedal which is used to sustain the sound of the active notes. For the robot hands, both left and right hands are Shadow Dexterous Hand [1] from the MuJoCo Menagerie [11].

C. System Architecture

1) Piano-playing Model and Goal: Piano playing is modeled by a finite-horizon MDP, defined by $(S, A, \rho, p, r, \gamma, H)$ where $S \subset \mathbb{R}^n$ is the state space, $A \subset \mathbb{R}^m$ is the action space, $\rho(\cdot)$ is the initial state distribution, $p(\cdot|s,a)$ controls the dynamics, $r : S \times A \to R$ is the rewards, $\gamma \in [0, 1)$ is the discount factor, and H is the horizon. The goal of an agent is to maximize the total expected discounted reward over the horizon or $\mathbb{E}\left[\sum_{t=0}^{H} \gamma r(s_t, a_t)\right]$.

2) Observation and Action Spaces: In the observation space, there are two types of observations in this study including proprioceptive which consists of hand and forearm joints, piano key joints and piano sustain pedal state, and goal state information that consists of a discrete vector of active fingers indicating which fingers should be used at that time step and a vector of piano key and sustain pedal goal states. A vector of a piano is stacked for lookahead horizon L which represents the number of lookahead steps.

TABLE I
SUMMARY OF OBSERVATION SPACE

Observations	Unit	Size
Hand and forearm joints	rad	52
Piano key joints	rad	88
Piano sustain pedal	discrete	1
Active finger goal states	discrete	10
Piano key and sustain pedal goal states	discrete	$L \times 89$

For the action space, there are also two parts of actions including joint angles for hand and forearm, and scalar value for the sustain pedal.

TABLE II
SUMMARY OF ACTION SPACE

Actions	Unit	Size
Hand and forearm joints	rad	44
Sustain pedal	discrete	1

3) Rewards: Similar to Robopianist [12], we utilize five reward terms to train an agent:

(i) key press term r_{key} which encourages the agent to press the right keys and discourages from pressing the wrong keys

$$r_{key} = 0.5 \cdot \left(\frac{1}{K} \sum_{i}^{K} g\left(\left\| k_s^i - 1 \right\|_2 \right) \right) + 0.5 \cdot (1 - 1_{\text{false\_positive}}) \tag{1}$$

Where K is the number of keys needed to be pressed at the current timestep, k_s is normalized joint of key between 0 to 1, g is tolerance function from dm_control [9] library, $1_{\text{false\_positive}}$ is indicator function (1 if any key that should not be pressed creates a sound)

(ii) sustain term $r_{sustain}$ which encourages the agent to press the sustain pedal at the right time

$$r_{sustain} = g\left(\left\| s_g - s_c \right\|_2 \right) \tag{2}$$

Where s_g is a sustain goal state and s_c is a current sustain state

(iii) move finger to key term r_{finger} which encourages the active fingers to move close to the right keys at the current time step

$$r_{finger} = \frac{1}{K} \sum_{i}^{K} g \left\| p_f^i - p_k^i \right\|_2 \tag{3}$$

Where p_f is the Cartesian position of the finger and p_k is the Cartesian position of the center of the key's surface

(iv) forearm colliding term $r_{forearm}$, which encourages the agent to move both hands without colliding

$$r_{forearm} = 0.5_{\text{not\_colliding}} \tag{4}$$

Where $0.5_{\text{not\_colliding}}$ is the indicator function (0.5 if both left and right forearms are not collided)

(v) energy penalty term r_{energy} which penalizes high energy expenditure

$$r_{energy} = |\tau_j|^T |v_j| \tag{5}$$

Where τ_j is a vector of joint torques, and v_j is a vector of joint velocities

Then, the final reward function is $r_{total} = r_{key} + r_{finger} + r_{sustain} + r_{forearm} - 0.005 \cdot r_{energy}$.

4) RL Algorithm and Network Architecture: In this study, we use DroQ [4] which is a variant of the Soft-Actor Critic (SAC) algorithm [3]. Overall system architecture is shown in Figure 1.

Moreover, we designed Q and policy networks, inspired by the existing system for a piano-playing task, Robopianist. We observed that piano key and sustain pedal goal states in the observation space are sequential data while others are static data. However, the existing system parameterized both actor and critic by only MLP which does not take advantage of the musical patterns. Therefore, we design an RNN-MLP network

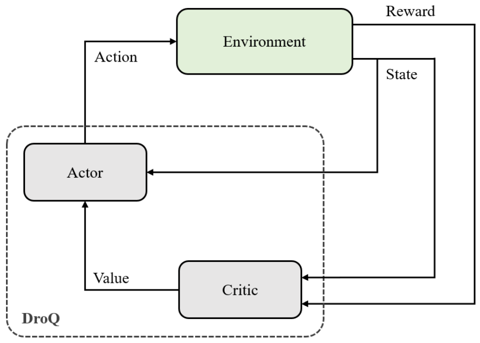

Fig. 1. Overall System Architecture

that can handle both sequential and static data for applying to Q and policy networks.

The RNN-MLP network consists of a single recurrent neural network (RNN) and three multilayer perceptrons (MLP). Sequential data from observations are inputs of RNN and then the outputs of RNN are concatenated with static data and fed to MLPs to create the desired outputs. The architecture of our RNN-MLP network is shown in Figure 2.

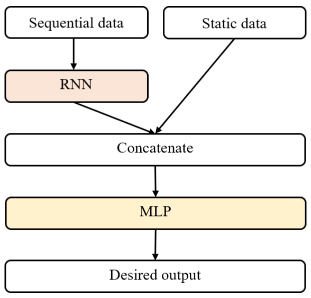

Fig. 2. Architecture of the RNN-MLP network

For a critic, we utilize a regularized variant of clipped double Q-learning similar to Robopianist, but we parameterized it by our RNN-MLP network with ReLU activations. Each MLP layer is followed by dropout with a rate of 0.01 and layer normalization. For an actor which is modeled by a tanh-diagonal Gaussian, we parameterized it by a RNN-MLP network that outputs mean and covariance.

In both actor and critic, RNN and MLPS have hidden layers with 256 neurons, their weights are initialized with Xavier initialization [2] and their biases are initialized to zero. For the training method, we followed the approach from Robopianist. First, 5000 observations with a uniform random policy are collected as warm-up steps before using a RL policy to sample actions and update an agent. Moreover, we also utilize Adam optimization with a learning rate of $3e^{-}4$ for both actor and critic, and a batch size of 256.

D. Baseline

We compare our model with the existing system that controls robot hands for a playing-piano task, named Robopianist [12]. It uses DroQ which is a variant of the Soft-Actor Critic (SAC) algorithm. For critic, they utilize a regularized variant of clipped double Q-learning where each Q-function is parameterized by a 3-layer MLP with ReLU activations and each MLP layer is followed by dropout with a rate of 0.01 and layer normalization. For the actor, it is modeled as a tanh-diagonal-Gaussian and also parameterized by a 3-layer MLP that outputs a mean and covariance.

E. Evaluation

We compared both piano notes and a sustain pedal from an agent with the ground truth and evaluated the agent's performance in a piano-playing task using precision, recall and F1-score in both piano notes and a sustain pedal every 10,000 time steps.

IV. Results

A. The Effectiveness of RNN

We investigate the effectiveness of RNN by comparing the agent with RNN-MLP network with the baseline using the same hyperparameters. The discount factor is set to 0.84 which is an effective discount factor, referenced from [12].

During training, as shown in Figure 3 the agent with RNN achieves the highest episode returns, outperforming all other models in terms of convergence efficiency and performance. When comparing policy learning curves from the agent with and without RNN, we observe that the agent with RNN-MLP network demonstrates better convergence efficiency compared to the baseline. Additionally, critic loss from the baseline increases and has higher variance at the end of the training steps while the agent with RNN-MLP network shows more stability and better learning trajectory.

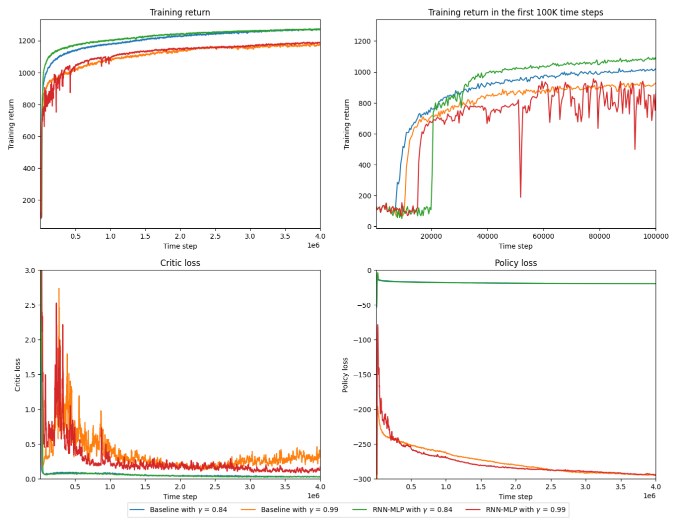

Fig. 3. (a) All episode returns and (b) episode returns in the first 100K time steps, (c) critic loss and (d) policy loss of the baseline and the agents with RNN-MLP network at the discount factor of 0.84 and 0.99 during training

However, after the policy converges, both models have a similar performance and the agent with RNN-MLP network requires more time steps to start learning compared to the baseline.

In addition, when evaluation, the F1-score does not align with the episode return. While the agent with RNN-MLP network shows better convergence efficiency, the baseline achieves 0.02 higher F1 score at the last time step.

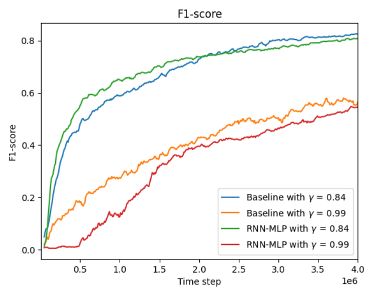

Fig. 4. F1-score of piano keys of the baseline and the agents with RNN-MLP network at the discount factor of 0.84 and 0.99 when evaluation

These results show the effectiveness of the RNN-MLP network for controlling a robot hand in the piano-playing task. While the RNN-MLP network is designed to capture the patterns in musical pieces, adding RNN layers can help the agent understand and extract these patterns. However, the RNN-MLP network has a drawback in computational efficiency that takes more time to train an agent.

B. The Effect of Discount Factor

RoboPianist [12] shows that the discount factor significantly affects F1-score. The discount factors that range from 0.85 to 0.92 produce the policies with the same performance, but high discount factors such as 0.99 lead to drop in the F1-score. However, we also investigate more about the effect of the discount factor, specifically on the RNN-MLP network, by increasing the discount factor from 0.84 to 0.99.

During training when using both MLP and RNN networks, we observe that the agents with the discount factor of 0.99 perform worse than the agents with the discount factor of 0.84. As shown in Figure 3, they have higher and noisy critic loss, showing instability in the learning process. Moreover, both critic and policy losses converge slower and the agents also achieve lower returns and policy loss when convergence.

In evaluation, as shown in Figure 4, we also observe that the agents with the discount factor of 0.99 show the worse performance, resulting in slower convergence and lower F1-score with 0.25 at the last timestep when using both MLP and RNN.

Moreover, an agent with RNN and the discount factor of 0.99 shows the worst performance compared to other agents. Its episode return and losses have high variance and the F1-score converges slowest among the agents.

In our analysis, we hypothesize that the rewards in the future are higher valued when using a high discount factor than using a lower discount factor. Therefore, high discount factors such as 0.99 are more suitable for the task that the rewards are in the future. However, in our task, the rewards are in every time step. Too high discount factors will lead an agent to too focus on future steps which are unnecessary and hard to learn, specifically for a long musical piece. Then, the results show higher losses, lower returns and also slower convergence of the agents with the discount factor of 0.99 compared to the agents with the discount factor of 0.84. Moreover, due to the drawback of RNN that takes more time to learn than MLP, an agent with RNN and the discount factor of 0.99 results in the worst performance during training.

To prove this hypothesis, we also decrease the discount factor to 0.69 and compare its results with the agents with the discount factor of 0.99. As shown in Figure 5, we observe that the performance of the agent with the discount factor of 0.69 is better than using the discount factor of 0.99. Critic and policy loss has lower variance and converge faster, proving that too high discount factors can drop the agent's learning ability.

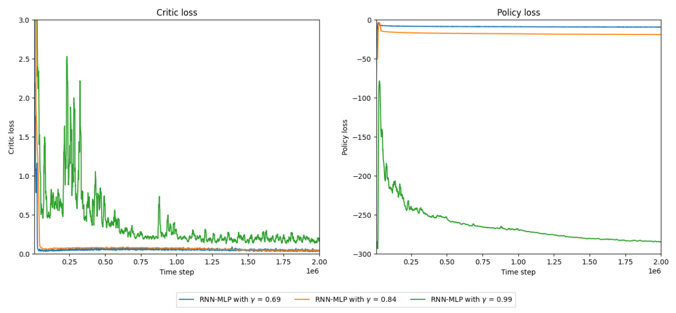

Fig. 5. (a) Critic loss and (b) policy loss of the agents with RNN-MLP network at the discount factor of 0.69, 0.84 and 0.99 in 2M time steps during training

However, the trend of F1-score from the agent with the discount factor of 0.69 seems to be lower than using the discount factor of 0.84. Since there is a limitation of time and the agent with the discount factor of 0.69 is trained only on 2M time steps which does not converge to optimal policy, we cannot summarize the significant difference between using the discount factor of 0.69 and 0.84 in this study.

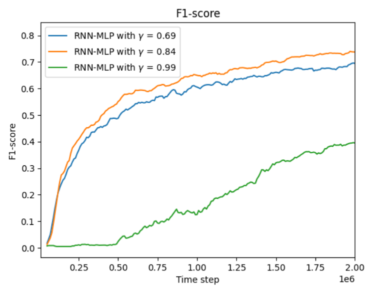

Fig. 6. F1-score of the agents with RNN-MLP network and the discount factor of 0.69, 0.84 and 0.99 in 2M time steps when evaluation

For the difference between policy losses when changing the discount factor, increasing the discount factors also increases the return, leading to lower policy loss which is calculated by Equation 6. Therefore, the agents with the discount factor of 0.99 have lower policy losses than the agents with the discount factor of 0.69 and 0.84.

$$\frac{1}{M}\sum_{i=0}^{M} Q(s,a) - \alpha \log \pi(a|s) \tag{6}$$

Where M is the number of Q functions

C. *Additional Results*

As described in IV, the agent with RNN and the discount factor of 0.84 shows the best performance in both training and evaluation. Therefore, we investigate the performance of this network architecture on other two musical pieces, Nocturne Op.9 No.2 and Polonaise Op.40 No.1 by Chopin, which have different characteristics from Für Elise. These pieces contain thirty-second notes, triplets and notes that are located far apart from each other. Moreover, Polonaise Op.40 No.1 also has multiple notes with a maximum number of notes of 9 at the same time step.

As shown in Figure 7, the agent still performs poorly on both Nocturne Op.9 No.2 and Polonaise Op.40 No.1 even in 5M time steps. They struggle with playing multiple notes at the same time and notes that are far apart from each other such as moving from D#4 to G1 with a single hand. However, precision, recall and F1-score do not converge, showing that the agents need to be trained more. Since there is a limitation of time that takes more than 48 hours for 5M time steps in both pieces, we cannot continue training the agents to improve their policies.

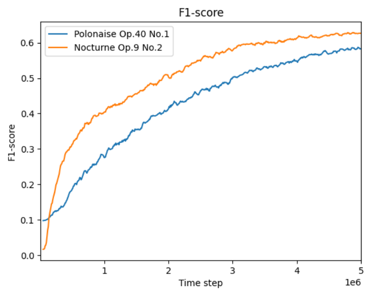

Fig. 7. F1-score of the agent with RNN-MLP network and the discount factor of 0.84 on Nocturne Op.9 No.2 and Polonaise Op.40 No.1

V. Conclusion

In this study, we present the piano playing task with robot hands by leveraging the musical patterns and a combination of RL approach and RNN to make the agent recognize the input information, resulting in performance improvement.

A previous RL approach used only multi-layer perceptrons in both critic and policy networks to develop the agent for playing piano without utilizing the advantage of musical patterns. To improve this method, we demonstrate the RNN-MLP network which can handle both sequential and static data from the musical pieces. The agents with this network achieve higher returns and faster convergence. In addition, we also demonstrate the effect of the discount factor which too high discount factor can lead to drop in the agent's performance using either only MLP or RNN-MLP network. Unfortunately, due to the limitation of time, we cannot summarize how too low discount factors affect the agent in this study.

Moreover, the agent with RNN-MLP network still has a downside in higher parameters compared to the baseline since there are more added layers. The final results, precision, recall and F1-score, of the baseline and the agent with RNN-MLP network are also quite similar. This shows that the RNN-MLP network can only outperform the baseline in the initial steps, but not surpass the baseline when both of them converge to the optimal policy.

For future work, the experiment of the discount factor is required to investigate more about the effect of this hyper-parameter and how to select the suitable value. In addition, other recurrent networks, such as LSTM, GRU or self-attention mechanisms which are developed to address the limitations of traditional RNN, can also be used to investigate their improvement over RNN and develop other agents to achieve higher performance in piano-playing tasks.

VI. Acknowledgment

We would like to thank RoboPianist for the support in contributing a benchmark of high-dimensional control in piano-playing tasks, MuJoCo physics engine and MuJoCo Menagerie in providing a simulated environment and robot hands and PIG dataset for supporting the musical pieces and annotations.

References

[1] S. R. Company, 2005. URL https://www.shadowrobot.com/dexterous-hand-series/.

[2] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[3] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. Technical report, 2018.

[4] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022.

[5] K.M.Instruments. Kawai vertical piano regulation manual, 2019. URL https://kawaius.com/wp-content/uploads/2019/04/Kawai-Upright-Piano-Regulation-Manual.pdf.

[6] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, 2015.

[7] Eita Nakamura, Yasuyuki Saito, and Kazuyoshi Yoshii. Statistical learning and estimation of piano fingering, 2020.

[8] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[9] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, November 2020.

[10] Huazhe Xu, Yuping Luo, Shaoxiong Wang, Trevor Darrell, and Roberto Calandra. Towards learning to play piano with dexterous hands and touch, 2022.

[11] Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL http://github.com/google-deepmind/mujoco_menagerie.

[12] Kevin Zakka, Philipp Wu, Laura Smith, Nimrod Gileadi, Taylor Howell, Xue Bin Peng, Sumeet Singh, Yuval Tassa, Pete Florence, Andy Zeng, and Pieter Abbeel. Robopianist: Dexterous piano playing with deep reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2023.