A Model-Free Deep Reinforcement Learning Approach to Piano Fingering Generation

Ananda Phan Iman
AI Graduate School
Gwangju Insitute of Science and Technology
Gwangju, Korea
anandaphan@gm.gist.ac.kr

Chang Wook Ahn*
AI Graduate School
Gwangju Insitute of Science and Technology
Gwangju, Korea
cwan@gist.ac.kr

***Abstract*—Piano fingering is a personal process for pianists to determine the appropriate finger one should use to play a musical note. In this paper, we propose a novel reinforcement learning framework with deep reinforcement learning with music score as the environment. Four environments are constructed from different right-hand monophonic music scores from various eras, types, and forms of classical music. Given current hand position information as the state, the pianist agent has to learn to choose the optimum action finger. We also propose a reward function that uses the fingering difficulty rules and reformulates them to compute the maximum negative difficulty of a fingering combination. We aim to explore how each approach method performs in the piano fingering generation and to identify the optimum approach between off-policy and on-policy model-free deep reinforcement learning. The results demonstrated that the off-policy method outperformed the other in training and evaluation while solving the problem using the DQN agent. In addition, the experiment showed a promising result for the future fingering generation without human supervision.**

***Index Terms*—piano fingering, symbolic music processing, deep q-network, reinforcement learning**

I. Introduction

Piano fingerings are often added to indicate the finger a pianist should use to play each note. Fingering of a musical piece has often been claimed as a major concern for the pianist. It may and does determine important technical and expressive elements of a performance [1]. Hence, determining fingering strategies has always been an intense interest of keyboard players because it is believed that fingering can significantly affect the technical and expressive qualities of a performance [2], [3]. Moreover, the process of fingering a piece is highly personal for each pianist [4] and depends on several factors such as the configuration of the keyboard and the physiology of the pianist's hand [1].

Finding the best fingering for piano pieces can be classified as learning by trial and error, [5]. This process allows the discovery of new skills and ways of doing things repetitively with various attempts. In this case, fingering skills are acquired by eliminating the unsatisfactory solution and continuously repeating to find the appropriate fingering for a piano passage.

Several methods in automatic piano fingering have been proposed to solve the piano fingering generation problem, including dynamic programming [6], local search algorithm [7], hidden markov model [8], [9]. However, this paper formulates the problem as a fully markov decision process(MDP) and approach with model-free deep RL.

The concept of reinforcement learning is that given an environment, an agent has to interact and try a variety of actions to progressively adapt and maximize the rewards [10]. In particular, it is concerned with how agents take sequential action and learn from the feedback from the environment. A reinforcement learning environment can be described with a markov decision process(MDP) consisting of a set of states, actions, a transition matrix, and rewards. Moreover, a model is referred to as a model of the environment that mimics the behavior of the environment, and model-free is the method that does not need any environment's model.

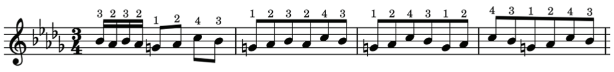

Fig. 1. Piano fingering of first 4 bar of Chopin waltz op.64 no.1 with DQN

The approach lets the agent practice the piano repeatedly from the environment and learn how to use the finger by following the rules. then the fingering is evaluated with our reward process called negative fingering difficulty where the agent has to find the most efficient fingering combination. Furthermore, this method enables reinforcement learning to approach piano fingering. Therefore in this paper, we aim to investigate how each basic method of RL behaves in our formulation and find out the best approach between off-policy and on-policy model-free deep reinforcement learning on solving piano fingering generation. The significant contributions of this work are threefold:

1) We create a framework for approaching piano fingering with deep reinforcement learning by using hand position information as a state.
2) We formulate the reward function by calculating the maximum *negative-difficulty* of a fingering combination.
3) We investigate the behavior of off-policy and on-policy methods. Our experiments show the off-policy performs better and the complex architecture of DDQN does not significantly improve the result of fingering.

* Corresponding author: Chang Wook Ahn

II. Background

A. Deep Reinforcement Learning

The action-value $Q_\pi(s,a) = \mathbb{E}[R_t \mid s,a]$ is defined as the expected return for selecting action a on state s under π [10]. The optimal action-value is denoted as $Q_*(s,a) = max_\pi Q_\pi(s,a)$. Then, the optimal policy π^* can be derived by selecting the highest valued action in each state. Similarly, the state-value function s for policy π is defined as $V_\pi(s) = \mathbb{E}[R_t \mid s]$ which represents the expected return of following π from state s. In the off-policy method, [11] firstly demonstrate the approach of deep q-network(DQN) algorithm to approximate Q value using neural network, which later stabilized in [12] using experience replay and the target network. Double deep q-network(DDQN) was later proposed by [13] to reduce overestimation of the DQN by decomposing max operation to action selection and action evaluation.

In contrast, the on-policy directly parameterized the policy $\pi(a \mid s;\theta)$ and updated the parameter θ by performing gradient ascend on expected return $\mathbb{E}(R_t)$. A baseline $b_t(s_t)$ is any function that reduces the variance by subtracting the expected return, called REINFORCE with baseline. [14]. One natural choice for the baseline is an estimate of the state value $b_t(s_t) \approx V(s_t;w)$ proposed by [15]. The update of parameter θ then defined as $\nabla_\theta \log \pi(a_t \mid s_t;\theta)(G_t - V(s_t;w))$. When the bootstrapping is implemented, the return of $G_t - V(s_t;w)$ can be seen as the advantage function $A(s_t,a_t) = Q(s_t,a_t) - V(s_t;w)$ where $Q(s_t,a_t)$ is critic that approximate action-state $Q(s_t,a_t) \approx Q_\pi(s,a)$. This approach is defined as the advantage actor-critic method (A2C) where the policy π is the actor and the baseline b_t is the critic. [16]

B. Piano Fingering Difficulty

Piano fingering difficulty was introduced in [17]. The fingering difficulty score is based on anatomic distance and finger motor constraints, where each finger pair has six different types of distance, namely MaxPrac, MinPrac, MaxComf, MinComf, MaxRel, MinRel.

TABLE I
Distance matrix of right-hand finger pair based on [17]

Pair	MinPrac	MinComf	MinRel	MaxRel	MaxComf	MaxPrac
1-2	-5	-3	1	5	8	10
1-3	-4	-2	3	7	10	12
1-4	-3	-1	5	9	12	14
1-5	-1	1	7	10	13	15
2-3	1	1	1	2	3	5
2-4	1	1	3	4	5	7
2-5	2	2	5	6	8	10
3-4	1	1	1	2	2	4
3-5	1	1	3	4	5	7
4-5	1	1	1	4	3	5

MaxPrac defined the maximum practical distance for two finger pairs to be stretched. MaxComf is the maximum stretch between two fingers that may be played without any noticeable effort. MaxRel is the maximum distance between two fingers when the finger is completely relaxed. MinPrac, MinComf, and MinRel can be defined similarly. Then the fingering difficulty score is given by summing the difficulty scores obtained by applying each of the 12 fingering rules specified. Lower fingering difficulty means easier to play.

C. Related Work

[8] proposed a hidden markov model (HMM) on right-hand only piano pieces. They represent the position of hand as one of the HMM states. [18] and [9] further extend the HMM model to deal with both fingering hands. They also constructed two feedforward networks and LSTM using pitch sequence and fingering with a sequence of integer pitches as the input and fingering number as the output. The result shows lower performance in match rate than HMM approach.

Ergonomic constraint-based method [17] was proposed to estimate piano fingering difficulty using the rules defined based on the finger distance. The approach is tested on the right hand only 7 opening fragments of Czerny pieces. They discovered that the fingering generated by this method is often similar to human-annotated fingering. The early experiment of the piano fingering difficulty model is applied by [19]. They conduct experiments to find out the determinant finger usage between beginner, intermediate, and advanced human pianists on the same piano pieces. [20] further extend the constraint-based method to estimate polyphonic music on both hands using the tabu search algorithm. They further proposed the variable neighborhood search(VNS) algorithm to handle the no improvement in the local optimum solution. [7].

The first method of approaching piano fingering generation as a fully markov decision process problem was proposed by [6]. They proposed a dynamic programming (DP) approach to monophonic piano pieces with the cost-based method. The cost of fingering is formulated as the sum of local costs for playing a pitch pair defined for each finger pair. Then, [21] use the DP approach to search the space of all possible fingering by proposing horizontal and vertical cost functions of [17] to solve polyphonic pieces.

III. Proposed Method

A. Pianist Agent

The goal of a pianist agent is to find the optimum finger given a specific piano piece. The agent is designed to learn how to choose the appropriate finger based on the current state of the music score. The fingers agent has to learn is numbered from 1 to 5. The thumb is denoted as 1, the index finger is 2, the middle finger is 3, the ring finger is 4, and the little finger is 5, as shown in figure 2. The finger chosen by the agent at each state must be the most comfortable, efficient, and well-compatible with the current hand position.

B. Environment Definition

We developed a custom environment called *PianoFingering* using gym library to learn the right-hand fingering of a music sheet. *PianoFingering* environment consists of four different pieces. Specifically, we chose one piece from Beyer

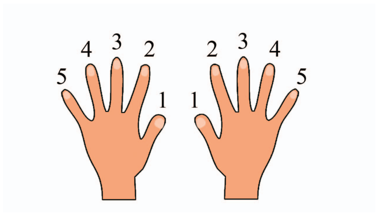

Fig. 2. Piano finger numbers

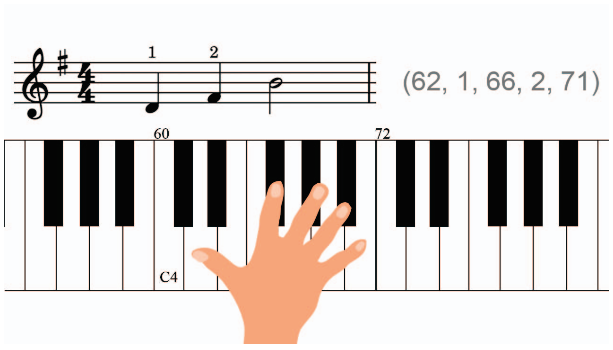

Fig. 3. Illustration of piano fingering as a reinforcement learning problem

Op.101 [22] and 3 pieces from Piano Fingering Dataset [9]. We considered three things in choosing the pieces: the era of classical music, the musical form, consists only of the monophonic right hand.

PianoFingering-V0 was created based on Beyer Op 101 No 12. *Vorschule im Klavierspiel*. Since it has a major influence on piano pedagogy as the manual for teaching students in the beginning stage, we adopt a similar concept by making version 0 the basic environment. The purpose of this environment is to check whether the designed algorithm performs well in the beginner piece. *PianoFingering-V1* was constructed based on Chopin Waltz Op.64 No. 1. This environment represents music from the romantic era, waltz form, and 3/4 time signature, *PianoFingering-V2* based on Beethoven For Elise, representing the classical era, rondo form, and 4/4 time signature, and *PianoFingering-V3* developed based on Scarlatti Sonata in A Major K.208, representing the baroque era, sonata form, and, 4/4 time signature.

Given a state s_t from the environment at time t, an agent has to select an action finger a_t. The quality of the chosen finger will be evaluated by the environment according to the reward function $R_t(s, a)$, and the agent will obtain the information of the next state s_{t+1}. The details of these components are described below.

1) State and Action: We defined the state space by the hand position information. The hand position can retrieved with the finger-action set of the past two notes and the current note,

Let n_t be the MIDI number of a note played at t, and a_t is the action finger taken at time t, then the state s_t is defined as

$$s_t = (n_{t-2}, a_{t-2}, n_{t-1}, a_{t-1}, n_t) \tag{1}$$

with $n_t \in [0, 127], n_t \in \mathbb{N}$. We then defined the action a_t as choosing a single action finger given the current state of hand, $a_t \in [1, 5], a_t \in \mathbb{N}$. Figure 3 illustrates the hand position of a bar of music. From the music sheet, D4 played with finger 1, F#4 played with finger 2, and B4 fingering is unknown. Then, the state s_t at time t can be defined as $(62,\ 1,\ 66,\ 2,\ 71)$ and agent has to choose the action finger a_t to play on B4.

2) Reward Function: Using the distance matrix described in Table I, the quality of an action is measured with the set of 12 fingering rules defined by [17]. These rules are:

1) **Stretch Rule**: add 2 points of difficulty for each consecutive semitone that is lower than MinComf or exceeded MaxComf.
2) **Small-span rule**: for finger pairs including thumb, add 1 difficulty point for each consecutive semitone lower than MinRel. For other finger pairs, add 2 points for each semitone.
3) **Large-span rule**: for finger pairs including thumb, add 1 difficulty point for each consecutive semitone that exceeds MaxRel. For other finger pairs, add 2 points per each semitone.
4) **Position-Change-count rule**: add 1 point if the distance between the first and third notes is lower than MinComf or exceeds MaxComf. In addition, add 1 more point if the second note is between the first and third notes, played with thumb, and the distance between the first and third notes is lower than MinPrac or exceeds MaxPrac.
5) **Position-Change size rule**: add 1 point of difficulty for the distance between the first and third note semitone that is lower than MinComf or exceeds MaxComf.
6) **Weak finger rule**: add 1 difficulty point when finger 4 or finger 5 is used.
7) **Three-Four-Five rule**: add 1 difficulty point if fingers 3,4,5 and their combination occurs consecutively.
8) **Three to four rule**: add 1 difficulty point if finger 4 is used right after finger 3
9) **Four on black rule**: add 1 difficulty point if fingers 3 and 4 occur consecutively in any order with finger 3 on the white key and finger 4 on the black keys
10) **Thumb on black rule**: add 1 difficulty point if the black key is played with finger 1. Then if the next note is white, add 2 more difficulty points. Moreover, if the previous note is white, add 2 more difficulty points
11) **Five on black rule**: add 2 difficulty points if finger 5 is in black key and the previous note is white key. Furthermore, if the next note is a white key, add 2 more points to the difficulty point
12) **Thumb passing rule**: add 1 difficulty point if the thumb passes on the same level, i.e. white key to white keys. Moreover, add 3 more points if the lower note is a white key played with a finger other than finger 1 and the upper

note is a black key played with finger 1.

We then define the difficulty of the selected finger a_t evaluated on rule k as $d_k(s_t, a_t)$. Then the total difficulty score at time t, $D_t(s_t, a_t)$ can be defined as the total of all rules k, $k \in 1, 2..M$

$$D_t(s_t, a_t) = \sum_{k=1}^{M} d_k(s_t, a_t) \tag{2}$$

with $d_k(s_t, a_t) \geq 0$. The total fingering difficulty of a full score $D(s, a)$ is the sum of all the difficulty score $D_t(s_t, a_t)$ in equation 2 obtained at all time $t \in 1, 2, ..N$.

$$\begin{aligned} \min \quad D(s,a) &= \sum_{t=1}^{N} D_t(s_t, a_t) \\ &= \sum_{t=1}^{N}\sum_{k=1}^{M} d_k(s_t, a_t) \end{aligned} \tag{3}$$

The function $D(s, a)$ is considered a minimization problem where the goal is to find the least difficult fingering combination. However, since the goal of reinforcement learning is to maximize the reward, we modify the calculation of equation 3 to be a maximization problem by multiplying D_t with -1 as shown in equation 4.

$$\max \quad D(s,a) = \sum_{t=1}^{N} -D_t(s_t, a_t) \tag{4}$$

Therefore, the problem becomes a maximization problem and we then defined it as a reward function $R_{t+1}(s_t, a_t)$.

$$R_{t+1}(s_t, a_t) = -D_t(s_t, a_t) = -\sum_{k=1}^{M} d_k(s_t, a_t) \tag{5}$$

Since $d_k \geq 0$, the reward agent receives will always be a negative number, $R_{t+1} \leq 0$. Hence as a maximization problem, the agent goal is to find the combination that can maximize the *negative fingering difficulty* or find the most efficient fingering combination. The proof can be derived from the identity of two negative numbers. The algorithm is described in algorithm 1

Algorithm 1 *Negative Fingering Difficulty* Reward Process

Input: current state s_t, action taken a_t

```
 1: function STEP($s_t, a_t$)
 2:     Initiate $D_t$ = 0
 3:     Extract $n_{t-2}, a_{t-2}, n_{t-1}, a_{t-1}, n_t$ from $s_t$
 4:     for $k = 1, M$ do
 5:         calculate the difficulty $d_k(s_t, a_t)$ and to $D_t$
 6:     end for
 7:     $R_{t+1} = -D_t$
 8:     if $n_{t+1} \neq \varnothing$ then
 9:         Set $s_{t+1}$ as $(n_{t-1}, a_{t-1}, n_t, a_t, n_{t+1})$
10:     else
11:         Set $s_{t+1}$ as None
12:     end if
13: end function
```

Output: next state s_{t+1}, reward R_t

C. Neural Network Architecture

For each step, each information in s_t is transformed into a 1x273 vector since there are only 88 playable notes in piano from A0 to C8, the note information is converted into 1x88, and the action finger into a 1x5 vector. Then we combine the vectors by concatenating them into one single vector. Hence, the input for the neural network is a 1x273 vector. We define this process as a state encoding process. The component for each network is described below.

The state-action value $Q(s_t, a_t; \theta)$ network for DQN and DDQN components are: The network takes in 1x273 vector size input, three hidden layers consisting of a fully connected layer with 512,256, and 64 nodes and ReLU activation per layer, one output layer with 5 output node. The output is the approximation of the Q-value of taking an action.

The state value network $V(s_t; w)$ for Policy Gradient and the critic network for A2C method will accept the 1x273 state encoding vectors as inputs, three hidden layers consisting of a fully connected layer with 512,256, and 64 nodes and ReLU activation per layer, one output layer with 1 node. The policy network $\pi_\theta(s_t, a_t)$ component for PG and Actor are: The network takes in 1x273 vector size input, three hidden layers consisting of a fully connected layer with 512,256, and 64 nodes and ReLU activation per layer, one output layer with 5 output node, and finally a softmax activation at the last layer.

D. Evaluation Measurement

We use two matrices to evaluate the quality of the agent network, namely fingering difficulty level and match rate.

We define fingering difficulty level $L(s, a)$ as the average difficulty of a hand movement at one time as one piece consists of N time step from $t = 1$. The number of hand movements can be associated with the total number of steps in the environment, which is N movement.

$$\begin{aligned} L(s,a) &= \frac{1}{N}\ D(s,a) \\ &= \frac{1}{N}\sum_{t=1}^{N}\sum_{k=1}^{M} d_k(s_t, a_t) \end{aligned} \tag{6}$$

the function can also be understood as how difficult the movement of the hand with the finger combination is in general. One of the advantages of using this equation is we can compare the difficulty level generated for any piece without considering the length of the pieces.

We also use the Match Rate evaluation matrix defined by [9]. General match rate M_{gen} is defined as an average value of the match rate from each human-annotated fingering, indicating how closely the generated output agrees with all the human-annotated fingering. Highest match rate M_{high} focuses on the human-annotated fingering that has the highest match rate. Soft match rate M_{soft} calculates the number of a generated finger that matches at least one of the human-annotated fingering, and recombination match rate $Mrec$ is the match rate of the generated fingering with the recombination of the multiple human-annotated fingering. Then, for the piece that contains only one human-annotated fingering, we only calculate the general match rate M_{gen}. For the piece that has multiple human-annotated fingering, we calculate the M_{gen}, M_{high}, M_{soft}, and M_{rec} values.

IV. Experiments & Results

The training was run on a single Nvidia RTX 2080 GPU with 8GB memory. We trained each agent model for 500 episodes with a discount factor γ of 0.99, Adam optimizer with a learning rate α of 0.001, and mean squared error (MSE) as the loss function. In addition, for DQN and DDQN agents, the replay memory was set to store data of the latest 10000 agent steps, the sampling size or batch size of 32, and the target model updated every 100 steps.

TABLE II
Results: Comparative evaluation of piano fingering generation, the least difficult result over 10 runs

Method	Difficulty Level $L(s,a)$ ↓ V0	V1	V2	V3
DQN	**0.6207**	**1.3721**	**3.1814**	**2.5441**
DDQN	**0.6207**	2.5163	5.2647	3.25
PG	8.1034	10.712	14.478	11.294
A2C	2.7586	3.8930	7.0441	6.6176
FHMM1[1]	1.0697	2.3349	3.6029	2.8320
FHMM2[1]	**0.6207**	1.8884	3.3235	2.8028
FHMM3[1]	**0.6207**	2.2512	3.2941	2.9859
Human[2]	**0.6207**	1.8419	4.0441	2.9155

[1] HMMs model from [9]
[2] fingering evaluated with the rules specified.

We analyze the agent behavior during the training and the time comparison of each method. We also compare the result by comparing the difficulty level with previous work on the same pieces. The difficulty evaluation is presented in table II and IV. Match rate evaluation is presented in table V. The result of each piece is presented visually in figure 5, 6, 7, 8.

A. Agent Training

We first trained all models on PianoFingering-V0 to check the behavior of each agent on the beginner pieces. This piece has one optimal fingering solution set by the composer. The maximum reward gained in one episode is -18 points or the difficulty level minimum is 0.6207. The total training time was around 27 minutes in total.

Despite the fastest training time for this environment of PG agents, its performance does not match that of the other. Both DQN and DDQN agents can reach the optimal solution with DDQN more stable in general.

As shown in figure 4, all methods start with a similar reward at the beginning of the training process. As the episodes begin, the PG agent fails to learn the policy of the environment,

TABLE III
Training details on PianoFingering-V0

Method	Run Time (s)	$\hat{L}(s,a)\downarrow$	Best ↓
DQN	51.6 ± 0.84	1.2552 ± 0.905	0.6207
DDQN	55.7 ± 0.95	0.7000 ± 0.238	0.6207
PG	9.8 ± 0.42	9.7138 ± 1.267	8.1034
A2C	45.1 ± 0.57	3.7448 ± 2.759	2.7586

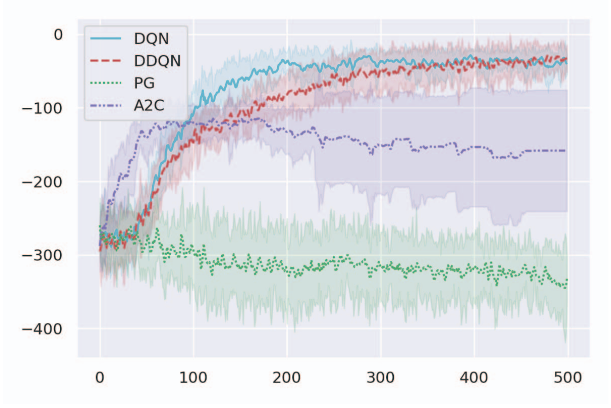

Fig. 4. Agent Training on Beginner Pieces

A2C agent shows a great learning process in the first hundred episodes but tends to achieve the local maximum at every run. In contrast, DQN and DDQN agents successfully learn the environment by getting the maximum reward of -18 points. Moreover, DQN agent could reach a maximum faster than DDQN in episodes 124 and 202, respectively.

In regards to the stability of the agent's learning process, the reward obtained per episode exhibits similar behavior as shown in Figure 4. The average rewards per episode of PG experiments resemble the initial prediction and tend to decrease with each run. However, the range for each episode remains consistent, indicating that the agent struggles to learn the policy and makes random predictions without improvement.

A2C agent performances show a fast learning process with a low standard deviation at early episodes. However, the learning stopped after reaching a certain point of the local maximum where the agent tries to do some exploration but fails to enhance the reward. On the other hand, both DQN and

TABLE IV
Training details of environment V1, V2, and V3

Method	V1 Time(s)	V1 $\hat{L}(s,a)\downarrow$	V2 Time(s)	V2 $\hat{L}(s,a)\downarrow$	V3 Time(s)	V3 $\hat{L}(s,a)\downarrow$
DQN	424	1.51 ± 0.17	294	3.78 ± 0.30	136	3.14 ± 0.53
DDQN	458	2.92 ± 0.21	288	6.05 ± 0.78	146	3.76 ± 0.35
PG	60	11.61 ± 0.39	43	16.5 ± 0.98	21	13.38 ± 1.05
A2C	345	5.08 ± 0.83	237	8.94 ± 1.33	115	7.75 ± 0.56

DDQN agents show stable performance in every environment. Their mean rewards per episode are increasing with a low standard deviation. It indicates both agents successfully learn the environment at every experiment.

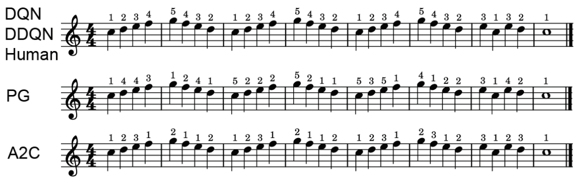

Fig. 5. Piano Fingering Result on Beyer Op.101 No.12.

B. Evaluation

Table IV shows the average difficulty level generated by the agent. The result indicates that the DQN agent can achieve a lower difficulty level than the other network. Even though the DDQN agent achieves more stability results in Environment V0, the result in the other version shows differently. Moreover, the on-policy approaches give relatively low maximum reward compared to off-policy approaches, with DQN outperforming the DDQN in every experiment. It implies that the complex architecture of DDQN does not provide any significant improvement for solving the formulated problem.

In Table II, we compare our best output of each agent with the previous work using HMMs [9]. The result shows that the reinforcement learning approach can get easier fingering difficulty than HMMs generally. We also evaluate our approach with match rate analysis detailed in table V.

TABLE V
MATCH RATE EVALUATION WITH HUMAN-ANNOTATED FINGERING

Method	Match Rate ↑			
	M_{gen}	M_{high}	M_{soft}	M_{rec}
DQN	**66.50%**	**68.97%**	78.52%	72.55%
DDQN	58.23%	61.10%	70.27%	62.29%
PG	24.20%	26.97%	34.60%	28.16%
A2C	43.08%	46.30%	54.18%	47.02%
FHMM1	63.50%	66.11%	**79.47%**	**74.22%**
FHMM2	61.32%	63.96%	76.61%	70.41%
FHMM3	60.62%	61.34%	75.42%	70.17%

The result in the table II, IV, and V shows that the PG and A2C agents have a high difficulty level and lower match rate with the human, indicating that the problem of piano fingering cannot be solved with on-policy reinforcement learning.

As shown in Table V, the DQN agent can get a higher general match rate and the highest match rate than the other method. It can be understood that DQN can generate more human-likely than the other method. Then, even though the soft match rate recombination match rate from DQN is lower than HMM order 1(FHMM1), the DQN agent can get a similar result to HMM order 1. This implies the DQN agent can capture the sequential consistency of fingering likewise the HMM order 1. It should be noted that reinforcement learning does not require any data from the ground truth.

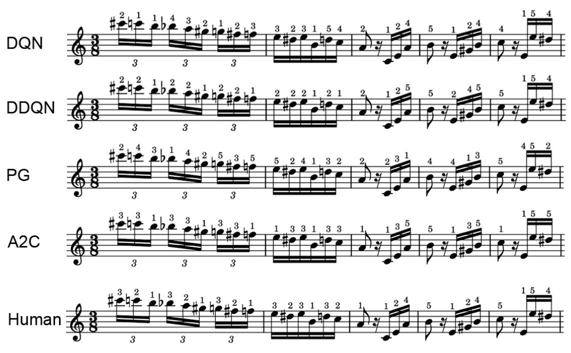

Fig. 6. Piano Fingering Result on Beethoven For Elise, shown only bar 81-85

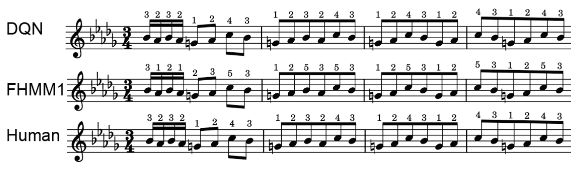

Fig. 7. Piano Fingering Result on first four bar of Chopin Waltz Op.64

C. Analysis of Fingering Generated

Figure 5 shows that the finger generated by DDQN and DQN agents can have an exact match as the optimal fingering. The fingering in the PG agent is random and A2C tends to avoid using Finger 4 and 5. In figure 6, the PG agent always randomly generates piano fingering. Similarly, the A2C agent avoids using weak fingers(Finger 4 and 5). while the DDQN agent uses 2-2-2-1-2-1-2 in E5-D#5-E5-D#5-B4 in bar 2, the DQN agent could use easier hand movement with 3-2-3-1-5-4-2. However, the usage of 1-5-4-2 is also considered hard given the short note distance from B4 to D5.

Notice that in figure 7, the result of the first four bars of DQN matches with human-annotated fingering. At the sequence of G4-Ab4-Bb4-Ab4-C5-Bb4, the DQN agent can learn to use 1-2-3-2-4-3, while FHMM1 avoids using finger 4 and use 1-2-5-3-5-3 instead, which is considered a relatively hard hand movement considering the note distance.

DQN and FHMM1 have relatively similar results in the first four bars of Scarlatti Sonata in figure 8,. Both can get similar results with the human-annotated fingering shown in the first and last bars. However, the use of fingering 5-4-3-2-1 in the second bar by FHMM1 is matched with human-annotated, which is easier to play than 4-3-1-3-2-1 by the DQN agent.

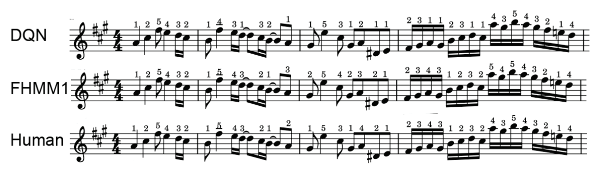

Fig. 8. Comparison on first four bars of Scarlatti Sonata in A Major K.208

D. Discussion and limitation

Our result demonstrates the potential of the model-free reinforcement learning approach on piano fingering problems. It should be noted that the method used in this experiment is the baseline DQN proposed by [12]. Thus, further off-policy or value-based reinforcement learning can be investigated in future experiments to solve the piano-fingering problem.

However, as our experiment is bound to right-hand only monophonic piano pieces, the formulation in solving polyphonic piano fingering can also be explored and the framework for solving both hands can be studied further. Moreover, as reinforcement learning relies on the reward function to give a good signal when the agent acts correctly, formulating a good indicator for piano fingering is still an open problem. Thus, our formulation of reward function as a maximization of negative difficulty can be used for exploring various rules of hand fingering to measure the difficulty level in the future.

V. Conclusion and Future Research

In this paper, we constructed an approach to solving the piano fingering generation problem using the model-free deep reinforcement learning method. We trained pianist agents to learn how to choose an optimum finger given a hand position in the music passage. we compared the performance of off-policy and on-policy methods, where DQN outperformed the other method. It showed that the complex architecture of DDQN does not provide a significant improvement to solve the problem. We compared it with the previous work and DQN could generate a relatively lower difficulty level than the other. When we evaluated with match rate analysis, DQN achieved a slightly better general match rate and high match rate than the other and predicted a similar result with HMM without any ground truth supervision.

In general, the off-policy methods performed better than the on-policy methods in solving piano fingering generation. It also indicated that finding the value of using one finger is more important than finding the probability of using a certain finger given a hand position. Additionally, as reinforcement learning depends on the reward function to provide a signal when the agent behaves appropriately, our proposed reward and evaluation calculation can be used to explore this topic with reinforcement learning in the future.

Acknowledgement

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01842, Artificial Intelligence Graduate School Program (GIST)) and the GIST Research Project grant funded by the GIST in 2024.

References

[1] A. Telles, *Piano fingering strategies as expressive and analytical tools for the performer*. Cambridge Scholars Publishing, 2021.

[2] J. Bamberger, "The musical significance of beethoven's fingerings in the piano sonatas," in *Music forum*, vol. 4, 1976, pp. 237–280.

[3] E. Clarke, R. Parncutt, M. Raekallio, and J. A. Sloboda, "Talking fingers: An interview study of pianists' views on fingering," *Musicae Scientiae*, vol. 1, pp. 107 – 87, 1997.

[4] L. Descaves, *Un Nouvel art du piano: exposés et documentation de pédagogie pianistique*. Fayard, 1966.

[5] K. Popper and W. W. Bartley III, *Realism and the aim of science: From the postscript to the logic of scientific discovery*. Routledge, 2013.

[6] M. Hart, R. Bosch, and E. Tsai, "Finding optimal piano fingerings," *The UMAP Journal*, vol. 21, no. 2, pp. 167–177, 2000.

[7] M. Balliauw, D. Herremans, D. P. Cuervo, and K. Sörensen, "A variable neighborhood search algorithm to generate piano fingerings for polyphonic sheet music," *Int. Trans. Oper. Res.*, vol. 24, pp. 509–535, 2017.

[8] Y. Yonebayashi, H. Kameoka, and S. Sagayama, "Automatic decision of piano fingering based on hidden markov models," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, p. 2915–2921.

[9] E. Nakamura, Y. Saito, and K. Yoshii, "Statistical learning and estimation of piano fingering," *ArXiv*, vol. abs/1904.10237, 2020.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013. [Online]. Available: https://arxiv.org/abs/1312.5602

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: http://arxiv.org/abs/1509.06461

[14] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[15] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, p. 229–256, 1992.

[16] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *2012 American Control Conference (ACC)*, 2012, pp. 2177–2182.

[17] R. Parncutt, J. Sloboda, E. Clarke, M. Raekallio, and P. Desain, "An ergonomic model of keyboard fingering for melodic fragments," *Music Perception - MUSIC PERCEPT*, vol. 14, pp. 341–381, 07 1997.

[18] E. Nakamura, N. Ono, and S. Sagayama, "Merged-output hmm for piano fingering of both hands," in *ISMIR*, 2014.

[19] J. Sloboda, E. Clarke, R. Parncutt, and M. Raekallio, "Determinants of finger choice in piano sight-reading," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 24, pp. 185–203, 02 1998.

[20] M. Balliauw, D. Herremans, D. P. Cuervo, and K. Sörensen, "A tabu search algorithm to generate piano fingerings for polyphonic sheet music," in *Proceedings of the International Conference on Mathematics and Computation in Music (MCM), London*, 2015.

[21] A. Kasimi, E. Nichols, and C. Raphael, "A simple algorithm for automatic generation of polyphonic piano fingerings." in *ISMIR*, 01 2007, pp. 355–356.

[22] F. Beyer, *Vorschule im Klavierspiel Op.101: Elementary method for the piano, op. 101*. Reprint in Alfred Publisher 2015, 1850.