RoboPianist: High-Dimensional Robotic Control of Bi-Dexterous Shadow Hands

CS224R Deep Reinforcement Learning

Pankhuri Aggarwal, Shelly Goel, Yoko Nagafuchi
Stanford University
pankhuri@stanford.edu, shelly23@stanford.edu, yokongf@stanford.edu

Abstract

In this project, we focus on training robotic control for a bi-manual high-dimensional task - playing the piano with shadow hands. We use two off-policy methods, namely Twin Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor-Critic (SAC). The objective of our project is to train agents to learn optimal policies for three bi-manual musical pieces of varying note patterns and complexity, namely - C major scale, D major scale, and Nocturne Rousseau. We integrate the RoboPianist and NanoRL implementations to simulate the piano, robotic hands, and musical elements using the Piano, Hand, and Sound RoboPianist models and the NanoRl framework. We also study the impact of different hyperparameters in optimizing the policy learning process, specifically fine-tuning pitch shifts and temporal shifts. Furthermore, we experiment with alternate key representations to reduce dimensionality and speed up the training process.

Our results show that TD3 outperforms SAC across all songs, with higher reward returns and faster convergence. Furthermore, we find that pitch shifts have no impact on performance but increasing temporal note stretches help nearly double reward returns. Finally, utilizing a smaller piano key representation range for certain songs, that span under three octaves, results in similar policy performance but with faster training speed and reduced computational costs. Overall, this research helps us advance the field of bi-dexterous robotic control. The insights of our research can be leveraged to ultimately improve other areas of robotic control for bi-manual dexterous tasks that rely on similar levels of precision and fluidity of robotic hand movements such as surgery and manufacturing.

1 Introduction

Piano-playing with shadow hands is a bi-manual multi-fingered robotic control task. It can arguably be thought of the ultimate challenge due to the required dexterity by the shadow hands, the complexity of the problem and the high-dimensionality of the problem space (Zakka, 2023) [1].

The objective of our project is to employ two off-policy methods to train robotic control for bi-dexterous piano-playing tasks. Zakka et al. introduced a training policy for dexterous high-dimensional control in precision, coordination, and planning. Our model integrates the Robopianist [2] and NanoRL [3] implementations. We focus on training agents to learn an optimal policy for specific musical pieces, including two-hand C major scale, two-hand D major scale, and two-hand Nocturne Rousseau. The two off-policy methods utilized in our research are Twin Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor-Critic (SAC). These methods are well-established algorithms for reinforcement learning and have been successfully applied to various control tasks. To further optimize performance, we explore tuning hyperparameters specifically, adjusting pitch shifts and temporal shifts. Our training process involves training the model on one song at a time. In our simulation, we utilize three models: the Piano model, which represents a standard 88-key digital piano; the Hand model, which consists of left and right Shadow Dexterous Hands sourced from Mujoco Menagerie; and the Sound model, which uses the Musical Instrument Digital Interface (MIDI) to represent musical pieces and synthesize sounds (Zakka, 2023) [1].

The potential impact of this project is advancing bi-dexterous piano-playing extends beyond the immediate application of being able to precisely and smoothly play any piano song. It can also enable us to gain invaluable insight into the field of robotic control for other bi-manual dexterous tasks, such as surgery or manufacturing since precision, smoothness, and hand coordination are integral to procedures in these fields as well.

In this paper, we provide a comprehensive overview of our research on bi-dexterous piano-playing robotic control. Then, we will cover the system design of the piano, hand, and sound models. We detail the reward function and the algorithms used. Next, we discuss our experiments and their results. We evaluate the performance of both TD3 and SAC agents in determining the optimal policy for each song. We also experiment with different temporal and pitch shifts to fine-tune the algorithms and alternative key representations to reduce dimensionality. Finally, we include limitations and possible extensions and conclude with a summary of our results and our analyses.

2 Related Work

We describe four related works on the robotic control of hands for tasks, including the original benchmarks introduced by Zakka et al. for bi-dexterous piano-playing.

Towards Learning to Play Piano with Dexterous Hands and Touch: Xu et al. [4] were the first to propose a reinforcement-learning (RL) based approach to the bi-dexterous task of learning to play the piano with robot hands. They presented this task as a Markov decision process (MDP) where actions are selected to minimize the difference between the actual key and the key played by the robot hand. To train a policy, they used the Soft-Actor Critic (SAC) algorithm and touch and audio-based reward functions. They successfully trained policy to effectively play simple yet diverse musical pieces. Through their work, they demonstrated the ability of an RL-based approach to enable robot hands to learn how to play the piano.

RoboPianist: Zakka et al. [1] introduced a new benchmark for the task of mastering the piano, which requires dexterous high-dimensional control in precision, coordination, and planning. The simulation consisted of a Piano model, Hand model using the left and right Shadow Hands from Mujoco, and Sound representation using Musical Instrument Digital Interface (MIDI). The reward is characterized by the incorporation of fingering information and the reward by multi-modal data, such as sensory and sound. The authors present two baseline methods. For the model-free method, the state-of-the-art off-policy algorithm DroQ is used, and each song is trained separately as a task. For the model-based method, model predictive control (MPC) is applied with Predictive Sampling.

BC-Z: In their paper, Jang et al. [5] present a zero-shot imitation learning system for the generalization of vision-based robotic manipulation systems toward unseen tasks, using both demonstrations and human interventions which we can explore in our project. In their study, to overcome the challenges with the generalization to multiple tasks, as opposed to objects, the authors employ scaling up high-quality data collection and conditioning the policy on the task specifications, such as a video of a person performing the task or providing language instructions.

Hybrid Learning: Pinosky et al. [6] proposed an approach to synthesize model-free and model-based methods for robots learning motor skills. They showed that their method performed better than other model-free, model-based, and combined approaches and was applicable to various learning approaches such as behaviour cloning and off-policy RL.

3 System Design

3.1 Simulation Setup

As set up by Zakka et al. (2023)[1], we simulate our RoboPianist model using the MuJoCo simulator[7]. We describe the model briefly in three aspects: the piano, hand, and sound models.

Piano Model The simulated piano is designed to closely align with an actual keyboard setup in terms of its dimensions, key positioning, and spacing. The model has 88 degrees of freedom, as it has the standard 88 keys that consist of 52 white and 36 black keys. As an extension to this project, we explored reducing the number of keys so that the number of parameters handled would be reduced for computational efficiency.

Hand Model The simulated hand model utilizes two Shadow Dexterous Hands from MuJoCo Menagerie [8]. It closely simulates the dexterousness of human hands with 20 degrees of freedom that include lateral arm motions as well as rotational wrist motions.

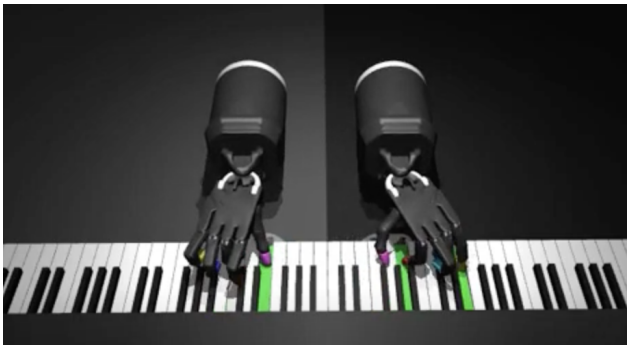


Figure 1: RoboPianist in action, showing the setup with keyboard and Shadow Hands.

Sound Model The sound input and piano pieces are represented with the Musical Instrument Digital Interface (MIDI) files. For the sound input, the files store four main features: binary values *note_on* and *note_off* depending on if a key is pressed on the piano, note pitches varying from 0 to 127, and note velocities varying from 0 to 127 that control note intensity.

3.2 Reward Structure

For this paper, we train a separate policy for every song and use the reward structure that incorporates hand and finger motions, following Zakka et al. (2023) [1]. The four components of the reward include key press (pressing the correct key and only the correct key), forearm penalty (minimizing forearm collisions), energy penalty (energy output of the hand actuators), and closeness of finger to the key. These are combined with different weights as detailed in Figure 2.

Reward	Formula	Weight	Explanation
Key Press	$0.5 \cdot g(\|\|k_s - k_g\|\|_2) + 0.5 \cdot (1 - \mathbf{1}_{\{\text{false positive}\}})$	1	Press the right keys and only the right keys
Forearm Penalty	$1 - \mathbf{1}_{\{\text{collision}\}}$	0.4	Minimize forearm collisions
Energy Penalty	$\lvert\tau_{\text{joints}}\rvert^\top \lvert v_{\text{joints}}\rvert$	-5e-3	Minimize energy expenditure
Finger Close to Key	$g(\|\|p_f - p_k\|\|_2)$	1	Shaped reward to bring fingers to key

Figure 2: This is the reward structure used for both algorithms following Zakka et. al (2023) [1]

The task of learning how to play the piano is formulated as a finite horizon Markov Decision Process (MDP), where an agent is to maximize its total expected reward in this finite horizon (Zakka, 2023) [1]. The total expected reward can be

formulated as:

$$\mathbb{E}[\sum_{t=0}^{H} \gamma^t r(s_t, a_t]$$

where H is the horizon, t is the time step, s_t and a_t are the state and action at time step and $\gamma \in [0, 1)$ is the discount factor, and r defines the rewards (Zakka, 2023) [1].

3.3 Algorithms

We apply two algorithms to train our high-dimensional robotic control task, namely Soft-Actor Critic and Twin Delayed Deep Deterministic Policy Gradients (TD3). These are chosen as they are both 'off-policy' methods and are able to learn efficiently from past samples using experience replay buffers [9]. Both methods are considered efficient and have their own advantages. These methods have been detailed below.

Soft-Actor Critic (SAC) SAC is an off-policy algorithm that can be used for environments with continuous action spaces. The SAC algorithm focuses primarily on entropy regularization. Entropy is a measure of randomness in the policy. Increasing entropy, increases exploration and prevents policy convergence to a local optimum. The policy maximizes the trade-off between expected return and entropy, which is similar to exploration-exploitation trade-off (OpenAI, 2018) [10].

SAC concurrently learns a policy π_θ and two Q-functions Q_{ϕ_1} and Q_{ϕ_2}. The loss functions for the Q-networks in SAC are:

$$L(\phi_i, \mathcal{D}) = (s, a, r, s', d) \sim \mathcal{D}\mathrm{E}\left[\left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2\right]$$

[10]

Twin Delayed Deep Deterministic Policy Gradients (TD3) TD3 is an off-policy algorithm for environments with continuous action spaces. It is a successor to the Deep Deterministic Policy Gradient (DDPG) and addresses the issue of policy breaking due to overestimation of Q-values, which is prevalent in DDPG. To achieve this, it introduces 3 features namely, clipped double-Q learning, delayed policy updates and target policy smoothing. They have shown to substantially improve performance over a DDPG (OpenAI, 2018) [11].

TD3 concurrently learns two Q-functions, Q_{ϕ_1} and Q_{ϕ_2}, by mean square Bellman error minimization. The policy is learned by maximizing Q_{ϕ_1}:

$$\max_\theta s \sim \mathcal{D}\mathrm{E}\left[Q_{\phi_1}(s, \mu_\theta(s))\right]$$

[11]

3.4 Dataset

To utilize the MIDI format as the sound and piano music input representations, we use generated MIDI files from the PIG dataset - a large-scale piano fingering dataset created by [12] which includes 150 piano pieces varying in genre and difficulty with fingering annotations by multiple pianists. Specifically, we use the 2-hand C Major scale, 2-hand D major scale, and Chopin Nocturne op. 9 no. 2 in E-flat Major with Rousseau's fingering (Nocturne Rousseau). To summarize, the input into the model is one MIDI file of one of the three songs which we call tasks, which contain information regarding the note and sound representations. The output of the model is the evaluation metrics from the training including returns and Q values which are described further below.

4 Experiments

We explored two types of off-policy algorithms, SAC and TD3, as well as hyperparameter fine-tuning by experimenting with different temporal and pitch shifts. Temporal shift refers to altering the timing of the song by speeding or slowing down the music. Pitch shifts refer to modifying the pitch of a note without modifying the duration by lowering it or raising the pitch by transposing a note to a different key.

We compared the performance of SAC and TD3 on three songs - two-hand C-Major Scale, two-hand D-Major Scale and Nocturne Rousseau. For hyperparameters, we experimented with temporal shifts $\in \{0.5, 2\}$ and pitch shifts $\in \{2, 3, 5, 7\}$.

Song	Model Type	Return	Q-Value	Actor Loss	Critic Loss
C Major Scale	SAC	266.02	248.51	-250.89	2.90
	TD3	419.56	214.61	-215.33	0.46
Nocturne Rousseau	SAC	1028.93	257.36	-261.45	8.04
	TD3	1617.39	190.03	-190.77	0.39
D Major Scale	SAC	248.34	211.20	-211.90	0.77
	TD3	421.63	258.49	-261.35	5.93

Table 1: Comparison of results of SAC and TD3 Models for C Major Scale, D Major Scale and Nocturne Rousseau (averaged over two runs).

Evaluation Metrics To compare the models, we report the returns, Q-values, actor loss and critic loss.

Hyperparameter Settings The model trains for a total of 50,000 steps. The optimizer used is ADAM with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 3×10^{-4}. The discount factor is 0.99 and minibatch size is 256. Other parameters are detailed in the Appendix (Table 4).

We based our implementation on the Nanorl and RoboPianist implementations by Zakka [3][2]. Additionally, all experiments were run on an AWS instance with 16 vCPUs.

4.1 Model Comparison

We compare both algorithms - SAC and TD3, for three songs - C Major scale, D Major scale and Nocturne Rousseau. We compare their returns, Q-values, actor loss and critic loss (averaged over two runs). The results are depicted in Table 1 and Figure 3.

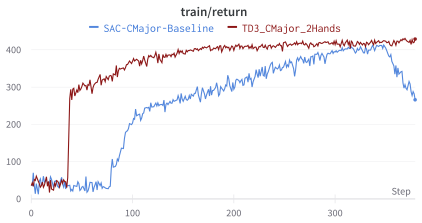
In general, the Nocturne Rousseau task is more complex than the C Major and D Major scale tasks. The C Major Scale task comprises of hitting notes of a scale, from C to the next C. In the Nocturne Rousseau task, we are not following a fixed pattern and there are variations in the sequence of the keys themselves that must be pressed. Unexpectedly, we had a higher reward return with the Nocturne Rousseau task compared to the C Major and D Major scale tasks. We believe this happens because the C Major and D Major scales are much shorter tasks with no repeating notes/sequences. A new note is pressed at each time step which makes it more complex for the policy to learn. However, Nocturne Rousseau is a longer musical piece with repeating notes and chords. Thus, the policy is able to perform better on it. Learning to play the piano is a complex task. The agent is trying to learn the correct finger positions, forearms positions, joint angles, notes etc. to play each chord/sequence. Thus, songs with repeating chords or sequences are likely to perform better especially, once the model has learnt how to play the chord/sequence correctly. Further experimentation with more songs can help to better understand what makes a song complex to learn for an agent.

In our results in Table 1, we also observe that TD3 is more effective than SAC on all three songs. It is evident that TD3 achieves higher returns with lower Q-values. Because the C Major and D Major scale tasks consist of notes in a sequence, TD3 which encourages exploration by utilizing twin critics and adding noise to the target action during training can help the algorithm discover better policies. Additionally, TD3 uses clipped double Q-learning to reduce overestimation bias, which can result in more accurate value estimation. The twin critics used in TD3 can provide a more reliable estimate of the value function, allowing the algorithm to learn faster and converge more efficiently. This can be observed in Figure 3. For all three songs, we can see that TD3 converges to high returns in much fewer steps than SAC. Additionally, TD3's delayed policy updates and avoiding over-estimating Q-values with the clipped double-Q learning can be advantageous for Nocturne Rousseau's more complex and variable key notes structure, leading to smoother and optimal policy update. Thus, TD3's success over SAC can be attributed utilization of twin critics for more exploration, delayed policy updates for a more complex task such as, Nocturne Rousseau, and avoiding over-estimation of Q-values.

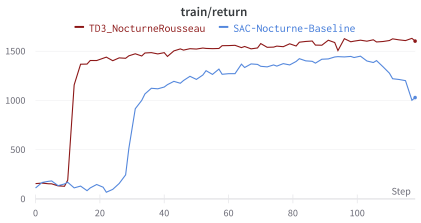
4.2 Comparison to Models with Temporal and Pitch Shifts

For hyperparameters, we experimented with temporal shifts $\in \{0.5, 1, 2\}$ and pitch shifts $\in \{0, 3, 7\}$, where pitch shift = 0 and temporal shift = 1 is the baseline. We use returns to compare the performance for these hyperparameters. The comparison is made across two songs - Nocturne Rousseau and C Major scale and for both algorithms.

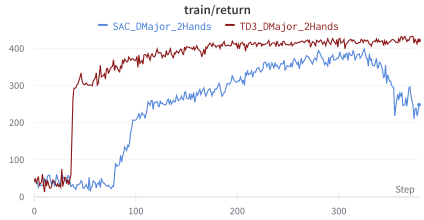
The results for experimentation with pitch shifts are depicted in Figures 4 and 5. The results for experimentation with temporal shifts are depicted in Figures 6 and 7.



(a) C Major Scale

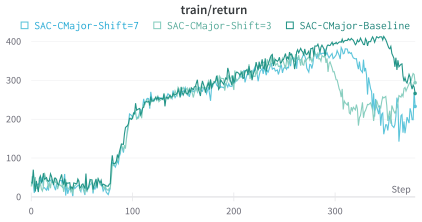


(b) Nocturne Rousseau

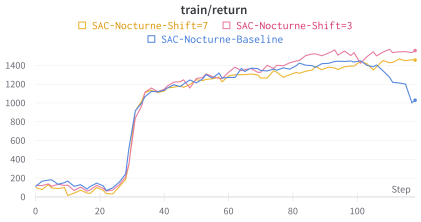


(c) Nocturne Rousseau

Figure 3: Comparing TD3 and SAC across three songs



(a) C Major Scale



(b) Nocturne Rousseau

Figure 4: Comparing Pitch Shifts for SAC

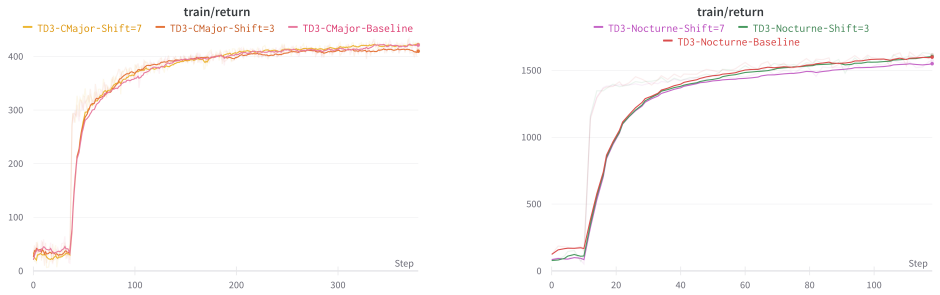
It can be seen from the results that pitch shift didn't yield noticeable improvements in the reward return. This could be because the relative motion of the hand is still the same even if we shift the pitch of the keys. However, having a larger temporal shift value almost doubled the reward return. This can be explained by how slowing the song by a factor of two allows our policy to spend more time per note, therefore being able to more effectively capture the patterns or specific notes that were a lot harder to reach at higher speeds.

4.3 Key Representations

The task of learning to play the piano using shadow hands is very high-dimensional, which makes it complex and computationally expensive. We experiment with using a reduced dimensional space for the task. For this, we experiment with key representations.

The original piano model key representation is for 88 keys. However, we observe that many songs, including our selected ones, spanned three octaves or less, meaning that a significant portion of the key range was not utilized. Therefore, we reduced the key representations to 36 keys, only utilizing the keys in the 3 central octaves. This helps to reduce dimensionality and speed up the training.

Since our previous experiments showed that TD3 outperformed SAC across all songs, we decided to focus on applying TD3 on two-hand D major scale. As seen in Figure 8, the training and reward returns in applying TD3 for two-hand D major scale is the same for the 36 key representation and the 88 key representation. Additionally, the training process becomes a lot faster.



(a) C Major Scale (b) Nocturne Rousseau

Figure 5: Comparing Pitch Shifts for TD3

(a) C Major Scale (b) Nocturne Rousseau

Figure 6: Comparing Temporal Shifts for SAC

This experiment highlights that we can reduce the key representation for certain songs that fall within a specific range of keys for faster training with same learning effectiveness. Thus, we are able to reduce dimensionality and reduce computational costs, without affecting performance.

5 Limitations and Future Work

Due to computational and time constraints, we experiment with a few different models on limited number of songs. We also work with a limited range of hyperparameters for finetuning. We encourage future work to focus on expanding experimentation to more songs and hyperparameters to further understand the impact of hyperparameters and SAC and TD3 on the task.

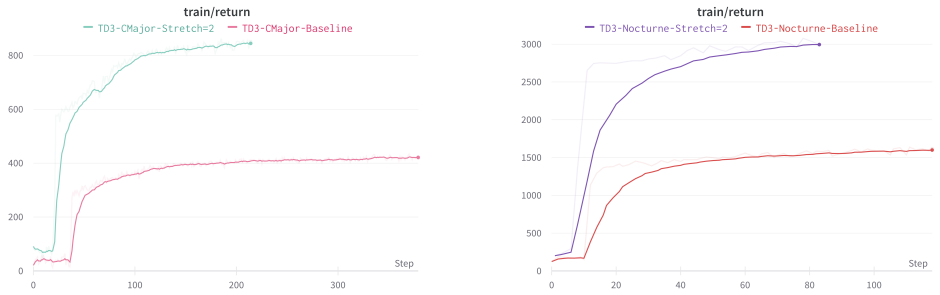
Additionally, we are only able to reduce key representation to 36 keys in this paper. Future work can focus on using 24 or even 12 keys (7 white and 5 black keys), to represent songs. This can be useful in reducing dimensionality for songs that use a limited range of notes.

Finally, our current work trains one policy per song. There is value in training policy that can learn a variety of songs. We hypothesize that learning the policy on multiple songs simultaneously could be used to generalize the learned policy on new songs with similar underlying features, such as finger and hand positioning and note patterns.

6 Conclusion

In this paper, we work on using deep reinforcement learning algorithms for the task of learning to play the piano with shadow hands. This is a complex and high-dimensional bi-dexterous robotic control task. We experiment with the benchmark - Robopianist, introduced by Zakka et al. (2023) [1].

We train one policy per song. The model uses a file with note and sound representations of a song as input. The reward function consists of four components - pressing the correct key, forearm collisions, energy penalty (energy output of the hand actuators), and closeness of finger to the key.



(a) C Major Scale (b) Nocturne Rousseau

Figure 7: Comparing Temporal Shifts for TD3

Song	Model	Pitch Shift	Temporal Shift	Return	Q-Value	Actor Loss	Critic Loss
C Major Scale	SAC	0	1.0	266.02	248.51	-250.89	2.90
	SAC	0	2.0	452.30	267.54	-271.01	11.37
	SAC	0	0.5	162.92	243.63	-245.02	0.75
	SAC	3	1.0	293.78	295.99	-297.59	2.55
	SAC	7	1.0	232.76	282.90	-286.01	9.79
	TD3	0	1.0	419.56	214.61	-215.33	0.46
	TD3	0	2.0	850.83	213.31	-214.09	0.52
	TD3	3	1.0	413.40	212.96	-213.70	0.48
	TD3	7	1.0	421.75	215.44	-216.12	0.47
Nocturne Rousseau	SAC	0	1.0	1028.93	257.36	-261.45	8.04
	SAC	0	2.0	2336.82	322.24	-325.33	3.20
	SAC	0	0.5	746.27	221.28	-222.29	0.43
	SAC	3	1.0	1558.17	226.89	-227.97	0.48
	SAC	7	1.0	1456.17	218.11	-219.10	0.44
	TD3	0	1.0	1617.39	190.03	-190.77	0.39
	TD3	0	2.0	2992.35	183.05	-183.78	0.38
	TD3	3	1.0	1615.95	192.09	-192.74	0.09
	TD3	7	1.0	1572.72	183.52	-184.20	0.1497

Table 2: Comparison of Training Scores of SAC and TD3 Models for C Major Scale and Nocturne Rousseau with temporal and pitch shifts.

We use two off-policy algorithms, namely - Soft Actor Critic and Twin Delayed Deep Deterministic Policy Gradients, for this task. We compare their performance across three songs - C Major scale, D Major scale and Nocturne Rousseau. TD3 outperforms SAC for all three songs. The policy converges to high returns in much fewer timesteps with TD3 than SAC. This can be attributed to TD3's use of twin critics to increase exploration. Additionally, TD3's use of clipped double Q-learning helps to reduce overestimation bias. The twin critics used in TD3 can provide a more reliable estimate of the value function, allowing the algorithm to learn faster and converge more efficiently.

Furthermore, we explore how fine-tuning hyperparameters such as augmenting the temporal and pitch shifts impact the learning performance with SAC and TD3. The pitch shift do not result in noticeable improvements or changes in the performance. However, larger temporal shifts helped nearly double the reward returns as the agent could focus on

Keys	Returns	Q-values	Actor Loss	Critic Loss
88	421.63	211.20	-211.90	0.78
36	426.52	211.63	-212.35	0.77

Table 3: Comparison for alternate key representation

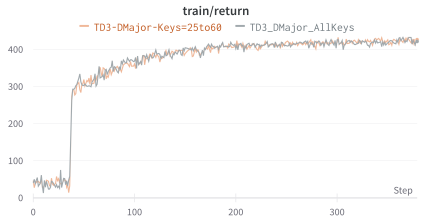


Figure 8: Comparison for alternate key representations

each note for double the time and understand underlying patterns to learn an optimal policy more effectively at slower speeds.

We also experiment with reducing the dimensionality of the task, by using alternate key representations. We experiment with using a representation with 36 keys instead of 88 keys. We find that for songs that have notes within a limited range, the results are not impacted but training speeds and computational efficiency increase.

Due to computational and time constraints, we are unable to experiment with multiple songs and key representations. We encourage future work in this direction.

References

[1] Kevin Zakka, Laura Smith, Nimrod Gileadi, Taylor Howell, Xue Bin Peng, Sumeet Singh, Yuval Tassa, Pete Florence, Andy Zeng, and Pieter Abbeel. Robopianist: A benchmark for high-dimensional robot control, 2023.

[2] Google Research. Robopianist. `https://github.com/google-research/robopianist`, 2023. June 12, 2023.

[3] Kevin Zakka. Nanorl. `https://github.com/kevinzakka/nanorl`, 2023. June 12, 2023.

[4] Huazhe Xu, Yuping Luo, Shaoxiong Wang, Trevor Darrell, and Roberto Calandra. Towards learning to play piano with dexterous hands and touch. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[5] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: zero-shot task generalization with robotic imitation learning. *CoRR*, abs/2202.02005, 2022.

[6] Allison Pinosky, Ian Abraham, Alexander Broad, Brenna Argall, and Todd D Murphey. Hybrid control for combining model-based and model-free reinforcement learning. *The International Journal of Robotics Research*, page 027836492210833, 2022.

[7] 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, 2012.

[8] DeepMind. Mujoco menagerie. `https://github.com/deepmind/mujoco_menagerie`, 2023. June 12, 2023.

[9] Vaishak V.Kumar. Soft actor-critic demystified, Jan 2019.

[10] OpenAI. Soft actor-critic, 2018.

[11] OpenAI. Twin delayed ddpg, 2018.

[12] Eita Nakamura, Yasuyuki Saito, and Kazuyoshi Yoshii. Statistical learning and estimation of piano fingering. *Information Sciences*, 517:68–85, 2020.

Appendix: Additional Information

Hyperparameter	Value
Total train steps	1M
Optimizer Type	ADAM
Learning rate	3×10^{-4}
β_1	0.9
β_2	0.999
Critic	
Hidden units	256
Hidden layers	3
Non-linearity	ReLU
Dropout rate	0.01
Actor	
Hidden units	256
Hidden layers	3
Non-linearity	ReLU
Misc.	
Discount factor	0.99
Minibatch size	256
Replay period	every 1 step
Eval period	every 10000 steps
Number of eval episodes	1
Replay buffer capacity	1M
Seed steps	5000
Critic target update frequency	1
Actor update frequency	1
Critic target EMA momentum (τ_Q)	0.005
Actor log std dev. bounds	[-20, 2]
Entropy temperature	1.0
Learnable temperature	True

Table 4: Hyperparameter Values (Zakka, 2023) [1]