

RP1M: A Large-Scale Motion Dataset for Piano Playing with Bi-Manual Dexterous Robot Hands

Supplementary Material

Anonymous Author(s)

Affiliation

Address

email

1 A RP1M Dataset Collection Details

2 A.1 Reward formulation

3 In Equation (3), we give the overall reward function used in our paper. We now give details of each
4 term. r_t^{Press} indicates whether the active keys are correctly pressed and inactive keys are not pressed.
5 We use the same implementation as [1], given as: $r_t^{\text{Press}} = 0.5 \cdot (\frac{1}{K} \sum_t^K g(\|k_s^i - 1\|_2)) + 0.5 \cdot (1 - \mathbf{1}_{\text{fp}})$.
6 K is the number of active keys, k_t^i is the normalized key states with range $[0, 1]$, where 0 means the
7 i -th key is not pressed and 1 means the key is pressed. g is tolerance from Tassa et al. [2], which is
8 similar to the one used in Equation (2). $\mathbf{1}_{\text{fp}}$ indicates whether the inactive keys are pressed, which
9 encourages the agent to avoid pressing keys that should not be pressed. r_t^{Sustain} encourages the agent
10 to press the pseudo sustain pedal at the right time, given as $r_t^{\text{Sustain}} = g(s_t - s_t^{\text{target}})$. s_t and s_t^{target} are
11 the state of current and target sustain pedal respectively. $r_t^{\text{Collision}}$ penalizes the agent from collision,
12 defined as $r_t^{\text{Collision}} = 1 - \mathbf{1}_{\text{collision}}$, where $\mathbf{1}_{\text{collision}}$ is 1 if collision happens and 0 otherwise. r_t^{Energy}
13 prioritizes energy-saving behavior. It is defined as $r_t^{\text{Energy}} = |\tau_{\text{joints}}| \tau |\mathbf{v}_{\text{joints}}|$. τ_{joints} and $\mathbf{v}_{\text{joints}}$ are joint
14 torques and joint velocities respectively.

15 A.2 Training details

16 **Observation Space** Our 1144-dimensional observation space includes the proprioceptive state of
17 dexterous robot hands and the piano as well as L -step goal states obtained from the MIDI file. In our
18 case, we include the current goal and 10-step future goals in the observation space ($L=11$). At each
19 time step, an 89-dimensional binary vector is used to represent the goal, where 88 dimensions are for
20 key states and the last dimension is for the sustain pedal. The dimension of each component in the
21 observation space is given in Table 1.

Table 1: Observation space.

| Observations | Dim |
|---------------------|--------------|
| Piano goal state | $L \cdot 88$ |
| Sustain goal state | $L \cdot 1$ |
| Piano key joints | 88 |
| Piano sustain state | 1 |
| Fingertip position | 30 |
| Hand state | 46 |

22 **Training Algorithm & Hyperparameters** Although our proposed method is compatible with
23 any reinforcement learning method, we choose the DroQ [3] as Zakka et al. [1] for fair comparison.

24 DroQ is a model-free RL method, which uses Dropout and Layer normalization in the Q function to
 25 improve sample efficiency. We list the main hyperparameters used in our RL training in 2.

Table 2: Hyperparameters used in our RL agent.

| Hyperparameter | Value |
|-----------------------|-----------------|
| Training steps | 8M |
| Episode length | 550 |
| Action repeat | 1 |
| Warm-up steps | 5k |
| Buffer size | 1M |
| Batch size | 256 |
| Update interval | 2 |
| Piano environment | |
| Lookahead steps | 10 |
| Gravity compensation | True |
| Control timestep | 0.05 |
| Stretch factor | 1.25 |
| Trim silence | True |
| Agent | |
| MLPs | [256, 256, 256] |
| Num. Q | 2 |
| Activation | GeLU |
| Dropout Rate | 0.01 |
| EMA momentum | 0.05 |
| Discount factor | 0.88 |
| Learnable temperature | True |
| Optimization | |
| Optimizer | Adam |
| Learning rate | 3e-4 |
| β_1 | 0.9 |
| β_2 | 0.999 |
| eps | 1e-8 |

26 A.3 Computational resources

27 We train our RL agents on the cluster equipped with AMD MI250X GPUs, 64 cores AMD EPYC
 28 “Trento” CPUs, and 64 GBs DDR4 memory. Each agent takes 21 hours to train. The overall data
 29 collection cost is roughly 21 hours * 2089 agents = 43,869 GPU hours.

30 A.4 MuJoCo XLA Implementation

31 To speed up training, we re-implement the RoboPianist environment with MuJoCo XLA (MJX),
 32 which supports simulation in parallel with GPUs. MJX has a slow performance with complex scenes
 33 with many contacts. To improve the simulation performance, we made the following modifications:

- 34 • We disable most of the contacts but only keep the contacts between fingers and piano keys
 35 as well as the contact between forearms.
- 36 • Primitive contact types are used whenever possible.
- 37 • The dimensionality of the contact space is set to 3.

38 • The maximal contact points are set to 20.
 39 • We use Newton solver with iterations=2 and ls_iterations=6.

40 After the above modifications, with 1024 parallel environments, the total steps per second is 159,376.

41 We use PPO implementation implemented with Jax to fully utilize the paralleled simulation. The PPO
 42 with MJX implementation is much faster than the DroQ implementation, which only takes 2 hours and
 43 7 minutes for 40M environment steps on the Twinkle Twinkle Little Star song while as a comparison,
 44 DroQ needs roughly 21 hours for 8M environment steps. However, the PPO implementation fails to
 45 achieve a comparable F1 score as the DroQ implementation as shown in Fig. 1. Therefore, we use
 46 the DroQ implement with the CPU version of the RoboPianist environment.

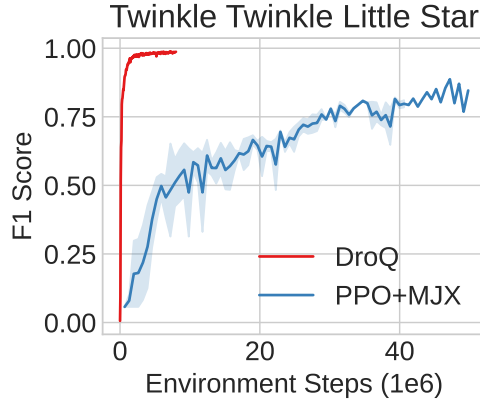


Figure 1: Comparison of the RL performance between DroQ and PPO with the MJX implementation of the RoboPianist environment. PPO+MJX is faster to run but has a worse performance than DroQ. We use DroQ with the CPU-version RoboPianist environment when training our RL agents.

47 **B Multitask Benchmarking Details**

48 A single multi-task policy capable of playing various songs is highly desirable. However, playing
 49 different music pieces on the piano results in diverse behaviors, creating a complex action distribution,
 50 particularly for dexterous robot hands with a large number of degrees of freedom (DoFs). This section
 51 introduces the baseline methods we have compared and the hyperparameters we have used. We also
 52 talk about the details of our multitask training and evaluation.

53 **B.1 Baselines and hyperparameters**

54 **B.1.1 BC**

55 Behavior Cloning (BC) [4] directly learns a policy by using supervised learning on observation-action
 56 pairs from expert demonstrations, which is one of the simplest methods to acquire robotic skills.
 57 Due to its straightforward approach and proven efficacy, BC is popular across multiple fields. The
 58 method employs a Multi-Layer Perceptron (MLP) as the policy network. Given expert trajectories,
 59 the policy network learns to replicate expert behavior by minimizing the Mean Squared Error (MSE)
 60 between predicted and actual expert actions. Despite its advantages, BC tends to perform poorly in
 61 generalizing to unseen states from the expert demonstrations. In our study, we evaluated three MLP
 62 models with varying hidden dimensions—256, 1024, and 4096. The first two models feature three
 63 layers, while the model with 4096 hidden dimensions is designed with six layers.

64 **B.1.2 IBC**

65 Implicit Behavioral Cloning (IBC) [5] adopts a novel angle on behavior cloning by reformulating
 66 supervised imitation learning as a conditional energy-based modeling problem. It trains an implicit

Table 3: BC

| Hyperparameter | Value |
|---------------------|-------|
| Batch Size | 256 |
| Optimizer | Adam |
| Learning Rate | 3e-4 |
| Activation | GELU |
| Training Steps | 1M |
| Observation Horizon | 1 |
| Prediction Horizon | 1 |
| Action Horizon | 1 |

67 policy represented by an energy function that is conditioned on both the action and observation,
 68 utilizing the InfoNCE loss [6]. This method demonstrates improved generalization over traditional
 69 BC. However, it encounters typical difficulties associated with training energy-based models, and the
 70 need for intensive action sampling and optimization at inference time, which may not scale well to
 71 high-dimensional action spaces.

Table 4: IBC

| Hyperparameter | Value |
|-----------------------------|--------|
| Batch Size | 256 |
| Optimizer | AdamW |
| Learning Rate | 1e-4 |
| Learning Rate Scheduler | cosine |
| Training Steps | 1M |
| Weight Decay | 1e-6 |
| Prediction Num of Iteration | 5 |
| Prediction Num of Sample | 1024 |
| Observation Horizon | 2 |
| Prediction Horizon | 2 |
| Action Horizon | 1 |

72 B.1.3 BC-RNN

73 BC-RNN [7] is a variant of BC that incorporates a Recurrent Neural Network as the policy network
 74 to capture a sequence of past observations. It is the best-performing baseline in the Robomimic
 75 paper [7].

Table 5: BC-RNN

| Hyperparameter | Value |
|-------------------------|--------|
| Batch Size | 256 |
| Optimizer | AdamW |
| Learning Rate | 1e-4 |
| Learning Rate Scheduler | linear |
| Training Steps | 1M |
| Observation Horizon | 1 |
| Prediction Horizon | 4 |
| Action Horizon | 1 |

76 B.1.4 Diffusion Policy

77 Diffusion models have achieved many state-of-the-art results across image, video, and 3D content
 78 generation [8, 9, 10, 11, 12]. In the context of robotics, diffusion models have been used as policy
 79 networks for imitation learning in both manipulation [13, 14, 15, 16] and locomotion tasks [17],
 80 showing remarkable performance across various robotic tasks. Diffusion Policy [13] proposed to
 81 learn an imitation learning policy with a conditional diffusion model. It models the action distribution
 82 by inverting a process that gradually adds noise to a sampled action sequence, conditioning on a state
 83 and a sampled noise vector. We used a CNN-based Diffusion Policy with DDIM [18] as the sampler
 84 to diffuse out action trajectories for improved efficiency. We build our diffusion policy training
 85 pipeline based on the Robomimic [7] and DROID [19], which provide high-quality implementations.

Table 6: Diffusion Policy

| Hyperparameter | Value |
|--------------------------|------------------|
| Batch Size | 128 |
| Optimizer | Adam |
| Learning Rate | 1e-4 |
| Learning Rate Scheduler | Linear |
| Training Steps | 1M |
| Diffusion Method | DDIM |
| EMA Power | 0.75 |
| U-Net Hidden Layer Sizes | [256, 512, 1024] |
| Observation Horizon | 2 |
| Prediction Horizon | 4 |
| Action Horizon | 1 |

86 B.2 Training and evaluation

87 We train the policies with 3 different sizes of expert data: 50, 150, and 300 songs, respectively.
 88 Subsequently, we assess the trained policies using three distinct categories of musical pieces. The
 89 first category, in-distribution songs, includes pieces that are part of the training datasets. Evaluating
 90 with in-distribution songs tests the multitasking abilities of the policies and checks if a policy can
 91 accurately recall the songs on which it was trained. The second group of songs for evaluation are
 92 easy out-of-distribution (OOD) songs: those music pieces do not overlap with the training songs
 93 but they are easy to play. They only contain slow motions and short horizons. The third group
 94 of evaluation songs are hard out-of-distribution songs: those are difficult music pieces that do not
 95 overlap with the training songs. They contain more diverse motions and longer horizons. This out-of-
 96 distribution evaluation measures the zero-shot generalization capabilities of the policies. Analogous
 97 to an experienced human pianist who can play new pieces at first sight, we aim to determine if it is
 98 feasible to develop a generalist agent capable of playing the piano under various conditions.

99 Additionally, our framework is designed with flexibility in mind, allowing users to select songs not
 100 included in our dataset for either training data collection or evaluation. Furthermore, users have the
 101 option to assess their policies on specific segments of a song rather than the entire piece.

Table 7: In-distribution songs

| |
|---|
| RoboPianist-etude-12-FrenchSuiteNo1Allemande-v0 |
| RoboPianist-etude-12-FrenchSuiteNo5Sarabande-v0 |
| RoboPianist-etude-12-PianoSonataD8451StMov-v0 |
| RoboPianist-etude-12-PartitaNo26-v0 |
| RoboPianist-etude-12-WaltzOp64No1-v0 |
| RoboPianist-etude-12-BagatelleOp3No4-v0 |
| RoboPianist-etude-12-KreislerianaOp16No8-v0 |
| RoboPianist-etude-12-FrenchSuiteNo5Gavotte-v0 |
| RoboPianist-etude-12-PianoSonataNo232NdMov-v0 |
| RoboPianist-etude-12-GolliwoggsCakewalk-v0 |
| RoboPianist-etude-12-PianoSonataNo21StMov-v0 |
| RoboPianist-etude-12-PianoSonataK279InCMajor1StMov-v0 |

Table 8: Easy out-of-distribution songs

| |
|---|
| RoboPianist-debug-TwinkleTwinkleLittleStar-v0 |
| RoboPianist-debug-CMajorChordProgressionTwoHands-v0 |
| RoboPianist-debug-TwinkleTwinkleRousseau-v0 |
| RoboPianist-debug-NocturneRousseau-v0 |
| RoboPianist-debug-NocturneRousseau-v0 |

Table 9: Hard out-of-distribution songs

| |
|---------------------------------------|
| GP-AkimenkoTheodoreAuCoinDuFeuOp28-v0 |
| GP-AgnewRoy2PianoPieces-v0 |
| GP-AlbaAntonioElEnsuenoOp16-v0 |
| GP-AlbaAntonioSensitiva-v0 |
| GP-MinotAdolfMisterioso-v0 |

References

- [1] K. Zakka, P. Wu, L. Smith, N. Gileadi, T. Howell, X. B. Peng, S. Singh, Y. Tassa, P. Florence, A. Zeng, et al. RoboPianist: Dexterous piano playing with deep reinforcement learning. In *7th Annual Conference on Robot Learning*, 2023.
- [2] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.
- [3] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka. Dropout Q-functions for doubly efficient reinforcement learning. *arXiv preprint arXiv:2110.02034*, 2021.
- [4] D. A. Pomerleau. Alvin: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.
- [5] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Thompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.
- [6] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [7] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1678–1690. PMLR, 08–11 Nov 2022. URL <https://proceedings.mlr.press/v164/mandlekar22a.html>.
- [8] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [9] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.
- [10] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.
- [11] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. DreamFusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.
- [12] Z. Liu, Y. Feng, M. J. Black, D. Nowrouzezahrai, L. Paull, and W. Liu. MeshDiffusion: Score-based generative 3d mesh modeling. *arXiv preprint arXiv:2303.08133*, 2023.
- [13] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.
- [14] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Conference on Robot Learning*, pages 3766–3777. PMLR, 2023.
- [15] M. Reuss, M. Li, X. Jia, and R. Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.
- [16] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

- 144 [17] X. Huang, Y. Chi, R. Wang, Z. Li, X. B. Peng, S. Shao, B. Nikolic, and K. Sreenath. Diffuseloco:
145 Real-time legged locomotion control with diffusion from offline datasets. *arXiv preprint*
146 *arXiv:2404.19264*, 2024.
- 147 [18] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint*
148 *arXiv:2010.02502*, 2020.
- 149 [19] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany,
150 M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma,
151 P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park,
152 I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat,
153 A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao,
154 J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung,
155 J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li,
156 K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O’Neill,
157 R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin,
158 Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman,
159 J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip,
160 Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation
161 dataset. 2024.