A The Details of MTDIFF

In this section, we give the pseudocodes of MTDIFF-P and MTDIFF-S in Alg. 1 and Alg. 2, respectively. Then we describe various details of the training process, architecture and hyperparameters:

- We set the per-task batch size as 8, so the total batch size is 400. We train our model using Adam optimizer [23] with $2e^{-4}$ learning rate for $2e^{6}$ train steps.
- We train MTDIFF on NVIDIA GeForce RTX 3080 for around 50 hours.
- We represent the noise model as the transformer-based architecture described in Section 3.3. MLP f_P which processes prompt is a 3-layered MLP (prepended by a layer norm [3] and with Mish activation). MLP f_{Ti} which processes diffusion timestep f_R is a 2-layered MLP (prepended by a Sinusoidal embedding and with Mish activation). f_R which processes conditioned Return and f_H which processes state history are 3-layered MLPs with Mish activation. f_A which processes actions, f_{TR} which process transitions and prediction head are 2-layered MLPs with Mish activation. The GPT2 transformer is configured as 6 hidden layers and 4 attention heads. The code of GPT2 is borrowed from `https://github.com/kzl/decision-transformer`.
- We choose the probability p of removing the conditioning information to be 0.25.
- In MTDIFF-P, we choose the state history length $L = 2$ for Meta-World and $L = 5$ for Maze2D.
- We choose the trajectory prompt length $J = 20$.
- We use $K = 200$ for diffusion steps.
- We set guidance scale $\alpha = 1.2$ for extracting near-optimal behavior.
- We choose $\beta = 0.5$ for low temperature sampling.

Algorithm 1 MTDIFF-P Training and Evaluation

 # Training Process

Initialize: training tasks $\mathcal{T}^{train}$, training iterations N, multi-task dataset $\mathcal{D}$, per-task batch size M, multi-task trajectory prompts Z

for $n = 1$ **to** N **do**
  for Each task $\mathcal{T}_i \in \mathcal{T}^{train}$ **do**
    Sample action sequences $\boldsymbol{x}_0^p(\tau_i)$ of length H and corresponding state history S_i^{prev} of length L from $\mathcal{D}_i$ with batch size M
    Compute normalized return $R(\tau_i)$ under τ_i
    Sample trajectory prompts τ_i^* of length J from Z_i with batch size M
  end for
  Get a batch $\mathcal{B} = \{\boldsymbol{x}_0^p(\tau_i), \tau_i^*, S_i^{\text{prev}}, R(\tau_i)\}_{i=1}^{|\mathcal{T}^{train}|}$
  Randomly sample a diffusion timestep $k \sim \mathcal{U}(1, K)$ and obtain noisy sequences $\boldsymbol{x}_k^p(\tau_i)$
  Omit the $R(\tau)$ condition with probability $\beta \sim \text{Bern}(p)$
  Compute $\mathcal{L}^p(\theta)$ and update MTDIFF-P model
end for

 # Evaluation Process

Given a task, reset the environment and set desired return $R_{\max}(\tau)$
Obtain the initial state history h_0, few-shot prompts Z
Set low-temperature sampling scale β, classifier-free guidance scale α
for $t = 0$ **to** $t_{\max}$ **do**
  Initialize $\boldsymbol{x}_K^p(\tau) \sim \mathcal{N}(\mathbf{0}, \beta \boldsymbol{I})$
  Sample $\tau^* \sim Z$, and formulate $\boldsymbol{y}'(\tau) = [h_t, \tau^*]$
  for $k = K$ **to** 1 **do**
    $\hat{\epsilon} = \epsilon_\theta\big(\boldsymbol{x}_k^p(\tau), \boldsymbol{y}'(\tau), \varnothing, k\big) + \alpha\big(\epsilon_\theta(\boldsymbol{x}_k^p(\tau), \boldsymbol{y}'(\tau), R_{\max}(\tau), k) - \epsilon_\theta(\boldsymbol{x}_k^p(\tau), \boldsymbol{y}'(\tau), \varnothing, k)\big)$
    $(\mu_{k-1}, \Sigma_{k-1}) \leftarrow \text{Denoise}(\boldsymbol{x}_k^p(\tau), \hat{\epsilon})$
    $\boldsymbol{x}_{k-1}^p(\tau) \sim \mathcal{N}(\mu_{k-1}, \beta\Sigma_{k-1})$
  end for
  Execute the first action from $\boldsymbol{x}_0^p(\tau)$ as the current action to interact with the environment
  Obtain the next state, and update h_t
end for

Algorithm 2 MTDIFF-S Training and Data Synthesis

 # Training Process

Initialize: training tasks $\mathcal{T}^{train}$, training iterations N, multi-task dataset $\mathcal{D}$, per-task batch size M, multi-task trajectory prompts Z

for $n = 1$ **to** N **do**
  for Each task $\mathcal{T}_i \in \mathcal{T}^{train}$ **do**
    Sample transition sequences $\boldsymbol{x}_0^s(\tau_i)$ of length H from $\mathcal{D}_i$ with batch size M
    Sample trajectory prompts τ_i^* of length J from Z_i with batch size M
  end for
  Get a batch $\mathcal{B} = \{\boldsymbol{x}_0^s(\tau_i), \tau_i^*\}_{i=1}^{|\mathcal{T}^{train}|}$
  Randomly sample a diffusion timestep $k \sim \mathcal{U}(1, K)$ and obtain noisy sequences $\boldsymbol{x}_k^s(\tau_i)$
  Compute $\mathcal{L}^s(\theta)$ and update MTDIFF-S model
end for

 # Data Synthesis Process

Initialize: synthetic dataset $\mathcal{D} = \varnothing$, synthesizing times M

Given a task, obtain few-shot prompts Z
for $m = 1$ **to** M **do**
  Initialize $\boldsymbol{x}_K^s(\tau) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  Sample $\tau^* \sim Z$, and formulate $\boldsymbol{y}_k^s(\tau) = [\tau^*]$
  for $k = K$ **to** 1 **do**
    $\hat{\epsilon} = \epsilon_\theta\big(\boldsymbol{x}_k^s(\tau), \boldsymbol{y}^s(\tau), k\big)$
    $(\mu_{k-1}, \Sigma_{k-1}) \leftarrow \text{Denoise}(\boldsymbol{x}_k^s(\tau), \hat{\epsilon})$
    $\boldsymbol{x}_{k-1}^s(\tau) \sim \mathcal{N}(\mu_{k-1}, \Sigma_{k-1})$
  end for
  Update $\mathcal{D} = \mathcal{D} \cup \boldsymbol{x}_0^s(\tau)$
end for

B The Details of Baselines

In this section, we describe the implementation details of the baselines:

- **PromptDT** uses the same prompts and GPT2 transformer in MTDIFF-P for taining. We borrow the code from `https://github.com/mxu34/prompt-dt` for implementation.
- **MTDT** embeds the tskaID which indicates the task to a embedding z with size 12, then z is concatenated with the raw state. With such conditioning, we broaden the original state space to equip DT with the ability to identify tasks in this multi-task setting. We keep other hyperparameters and implementation details the same as the official version `https://github.com/kzl/decision-transformer/`.
- **MTIQL** uses a multi-head critic network to predict the Q value for each task, and each head is parameterized with a 3-layered MLP (with Mish activation). The actor-network is parameterized with a 3-layered MLP (with Mish activation). During training and inference, the scalar taskID is embedded via 3-layered MLP (with Mish activation) into latent variable z, and the input of the actor becomes the concatenation of the original state and z. We build MTIQL based on the code `https://github.com/tinkoff-ai/CORL` [59].
- **MTCQL** is applied with a similar revision in MTIQL. The main difference is that MTCQL is based on CQL [25] algorithm instead of the IQL algorithm [24]. We build MTCQL based on the code `https://github.com/tinkoff-ai/CORL` [59].
- **MTBC** uses a similar taskID-cognitional actor in MTIQL and MTCQL. For training and inference, the scalar taskID is embedded via a 3-layered MLP (with Mish activation) into latent variable z, and the input of the actor becomes the concatenation of the original state and z. The actor is parameterized with a 3-layered MLP and outputs predicted actions.
- **RAD** adopts the random amplitude scaling [28] that multiplies a random variable to states during training, i.e., $s' = s \times z$, where $z \sim \text{Uniform}[\alpha, \beta]$. We choose $\alpha = 0.8$ and $\beta = 1.2$.

- **S4RL** adopts the adversarial state training [52] by taking gradients with respect to the value function to obtain a new state, i.e. $s' \leftarrow s + \epsilon \nabla_s \mathbb{J}_Q(\pi(s))$, where $\mathbb{J}_Q$ is the policy evaluation update performed via a Q function, and ϵ is the size of gradient steps. We choose $\epsilon = 0.01$.

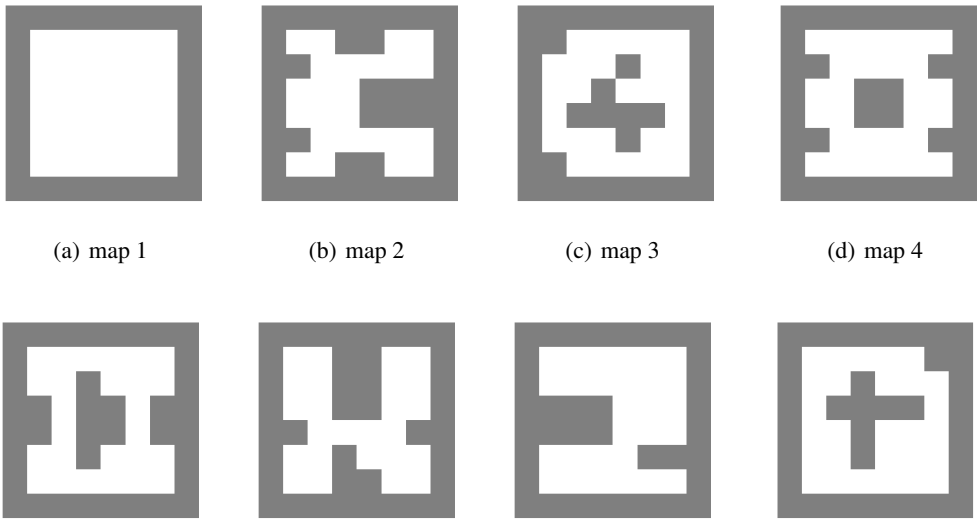
C Ablation Study on Model Architecture

The architecture described in §3.3 handles input types of different modalities as tokens that share similar formats, actively capturing interactions between modalities. The incorporation of a transformer is also helpful for sequential modeling. To ablate the effectiveness of our architecture design, we train MTDIFF-P using U-Net with a similar model size to ours on the near-optimal dataset. We use a Temporal Convolutional Network (TCN) [4] to encode the prompt into an embedding z_p, and inject it in the U-Net layers as a condition. We follow the conditional approach in [2] and borrow the code for temporal U-Net from `https://github.com/jannerm/diffuser` [21]. The results summarized in Table 3 show that our model architecture outperforms U-Net to learn from multi-task datasets.

Table 3: Average success rate across 3 different seeds of MTDIFF-P and MTDIFF-P (U-Net) on MT50-rand.

Methods	Success rate on near-optimal dataset (%)	Success rate on sub-optimal dataset (%)
MTDIFF-P	$\mathbf{59.53 \pm 1.12}$	$\mathbf{48.67 \pm 1.32}$
MTDIFF-P (U-Net)	55.67 ± 1.27	47.42 ± 0.74

D Environmental Details of Maze2D



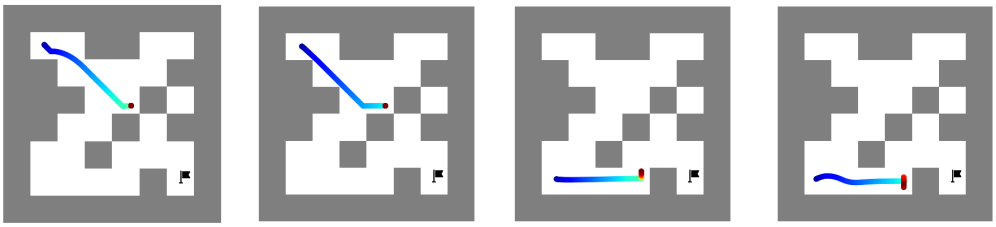
(a) map 1 (b) map 2 (c) map 3 (d) map 4

(e) map 5 (f) map 6 (g) map 7 (h) map 8

Figure 8: 2D visualization of eight training maps designed in Maze2D.

We design eight different training maps for multi-task training, which are shown at Fig. 8. Different tasks have different reward functions and transition functions. For generalizability evaluation, we have designed one new unseen map. Although in Fig. 4, MTDIFF is able to generalize on new maps while PromptDT fails, we should acknowledged that MTDIFF may fail at some difficult unseen cases, as shown in Fig. 9. The reasons may lie in 2 folds: One is the inherent difficulty of the case itself, and the other is that the case's deviation from the distribution of the training cases surpasses the upper threshold of generalizability of MTDIFF. We also provide another map where MTDIFF succeeds while PromptDT fails in Fig. 10.

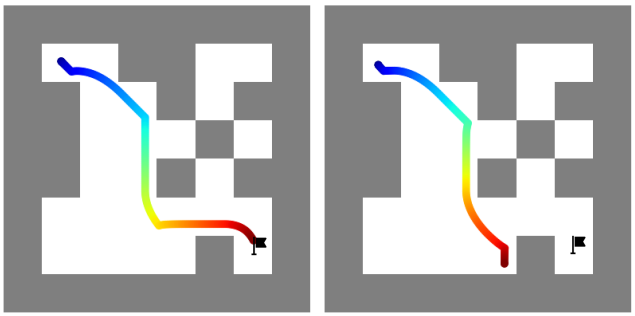
We collect 35k episodes together to train our model and PromptDT. The episodic length is set as 600 for training and 200 for evaluation. After training with 512 batch size for 2e5 gradient steps, we evaluate these methods on seen and unseen maps.



(a) PromptDT (b) MTDIFF (c) PromptDT (d) MTDIFF

(1) Failure case 1 **(2) Failure case 2**

Figure 9: 2D visualization of 2 difficult cases where both PromptDT and MTDIFF both fail. These 2 cases are both unseen during training. Goal position is denoted as .



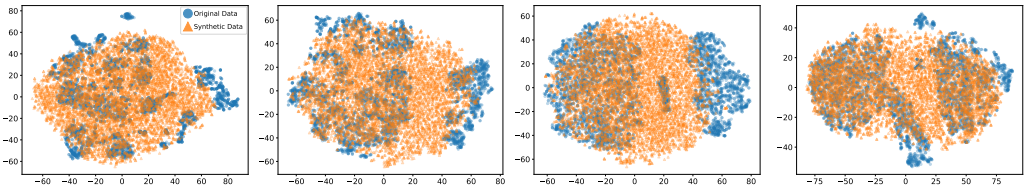
(a) MTDIFF-P (b) PromptDT (failed)

Figure 10: Unseen maps of Maze2D with long planning path. MTDIFF-P reach the goal while PromptDT fails. Goal position is denoted as .

E Data Analysis

E.1 Distribution Visualization

We find the synthetic data is high-fidelity, covering or even broadening the original data distribution, which makes the offline RL method performs better in the augmented dataset. The distribution visualization is shown in Fig. 11.



(a) Coffee-push (b) Disassemble (c) hand-insert (d) box-close

Figure 11: 2D visualization of sampled synthetic data and original data via T-SNE [61]. The data is sampled from tasks coffee-push, disassemble, hand-insert and box-close respectively.

E.2 Statistical Analysis

Following SynthER [38], we measure the Dynamics Error (MSE between the augmented next state and true next state), and L2 Distance from Dataset (minimum L2 distance of each datapoint from the dataset) for the augmented data of each method (i.e., MTDIFF-S, S4RL and RAD), as shown in Table 4. Since S4RL performs data augmentation by adding data within the ϵ-ball of the original data points, it has the smallest dynamics error with a small ϵ. RAD performs random amplitude scaling and causes the largest dynamics error. We remark that S4RL performs local data augmentation around the original data and can be limited in expanding the data coverage of offline datasets. In contrast, our method generates data via diffusion model without explicit constraints to the original data points, which also has small dynamics error and significantly improves the data coverage, benefiting the offline RL training.

Table 4: Comparing L2 distance from the training dataset and dynamics error under each method.

Methods	Dynamics Error	L2 Distance from Dataset
MTDIFF-S	0.0174	0.4552
S4RL	0.0001	0.4024
RAD	0.0641	0.4617

F Limitations and Discussions

In this section, we will discuss the limitations and broader impacts of our proposed method MTDIFF.

Limitation. Diffusion models are bottlenecked by their slow sampling speed, which caps the potential of MTDIFF for real-time control. How to trade off the sampling speed and generative quality remains to be a crucial research topic. For a concrete example in MetaWorld, it takes on average 1.9s in wall-clock time to generate one action sequence for planning (hardware being a 3090 GPU). We can improve the inference speed by leveraging a recent sampler called DPM-solver [37, 36] to decrease the diffusion steps required to $0.2\times$ without any loss in performance, and using a larger batch size (leveraging the parallel computing power of GPUs) to evaluate multiple environments at once. Thus the evaluation run-time roughly matches the run-time of non-diffusion algorithms (diffusion step is 1). In addition, consistency models [55] are recently proposed to support one-step and few-step generation, while the upper performance of what such models can achieve is still vague.

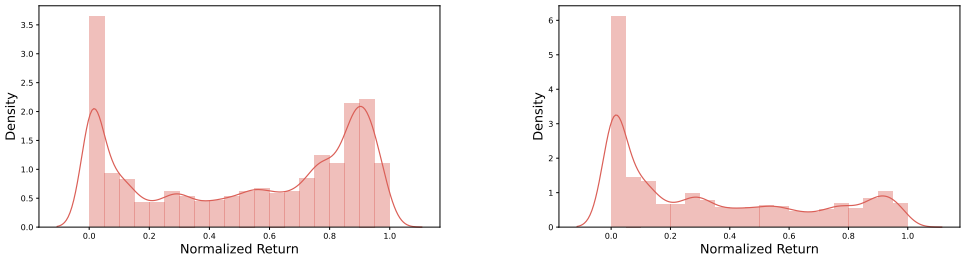
Broader Impacts. As far as we know, MTDIFF is the first proposed diffusion-based approach for multi-task reinforcement learning. It could be applied to multi-task decision-making, and also could be used to synthesize more data to boost policy improvement. MTDIFF provides a solution for achieving generalization in reinforcement learning.

G Dataset collection

Meta-World. We train Soft Actor-Critic (SAC) [18] policy in isolation for each task from scratch until convergence. Then we collect 1M transitions from the SAC replay buffer for each task, consisting of recording samples in the replay buffer observed during training until the policy reaches the convergence of performance. For this benchmark, we have two different dataset compositions:

- **Near-optimal** dataset consisting of the experience (100M transitions) from random to expert (convergence) in SAC-Replay.
- **Sub-optimal** dataset consisting of the initial 50% of the trajectories (50M transitions) of the near-optimal dataset for each task, where the proportion of expert data decreases a lot.

To visualize the optimality of each dataset clearly, we plot the univariate distribution of return in each kind of dataset in Fig. 12. Our dataset is available at `https://bit.ly/3MWf40w`.



(a) Near-optimal dataset (b) Sub-optimal dataset

Figure 12: Density visualization of the normalized return in the dataset.

Maze2D The offline dataset is collected by selecting random goal locations and using a planner to generate sequences of waypoints by following a PD controller. We borrow the code from `https://github.com/Farama-Foundation/D4RL` [15] to generate datasets for 8 training maps. We collect 35k episodes in total.

H Differences Between PromptDT and MTDIFF-P

The remarkable superiority of MTDIFF-P over PromptDT emerges from our elegant incorporation of transformer architecture and trajectory prompt within the diffusion model framework, effectively modeling the multi-task trajectory distribution. PromptDT is built on Decision Transformer and it is trained in an autoregressive manner, which is limited to predicting actions step by step. However, MTDIFF-P leverages the potency of sequence modeling, empowering it to perform trajectory generation adeptly. MTDIFF-P has demonstrated SOTA performance in both multi-task decision-making and data synthesis empirical experiments, while PromptDT fails to contribute to data synthesis. Technically, MTDIFF-P extends Decision Diffuser [2] into the multi-task scenario, utilizing classifier-free guidance for generative planning to yield high expected returns. To further verify our claim, we train our model on the publicly available PromptDT datasets [69], i.e., Cheetah-vel and Ant-dir. These chosen environments have been judiciously selected due to their inherent diversity of tasks, serving as a robust test to validate the capability of multi-task learning. We report the scores (mean and std for 3 seeds) in Table 5.

Table 5: Average scores obtained by MTDIFF-P and PromptDT across 3 seeds. We observed that MTDIFF-P outperforms PromptDT largely, demonstrating its high efficacy and potency.

Methods	Cheetah-vel	Ant-dir
MTDIFF-P	$\mathbf{-29.09 \pm 0.31}$	$\mathbf{602.17 \pm 1.68}$
PromptDT	$\mathbf{-34.43 \pm 2.33}$	$\mathbf{409.81 \pm 9.69}$

I Single-Task Performance

We train one MTDIFF-P model on MT50-rand and evaluate the performance for each task for 50 episodes. We report the average evaluated return in Table 6.

Table 6: Evaluated return of MTDIFF-P for each task in MT50-rand. We report the mean and standard deviation for 50 episodes for each task.

Tasks	Return on near-optimal dataset	Return on sub-optimal dataset
basketball-v2	2735.7 ± 1927.7	2762.7 ± 1928.8
bin-picking-v2	733.7 ± 1211.8	59.6 ± 32.3
button-press-topdown-v2	1491.6 ± 390.9	1395.8 ± 332.6
button-press-v2	2419.2 ± 413.7	2730.4 ± 514.3
button-press-wall-v2	3474.7 ± 887.6	2613.1 ± 906.8
coffee-button-v2	3157.9 ± 1274.3	1649.4 ± 869.5
coffee-pull-v2	437.5 ± 597.3	98.8 ± 115.4
coffee-push-v2	463.7 ± 729.1	71.3 ± 55.7
dial-turn-v2	2848.1 ± 996.3	2244.5 ± 881.5
disassemble-v2	369.9 ± 265.4	372.3 ± 623.2
door-close-v2	4325.2 ± 377.6	4270.4 ± 460.1
door-lock-v2	3215.1 ± 857.5	3082.4 ± 1041.4
door-open-v2	3458.1 ± 960.5	2457.0 ± 833.0
door-unlock-v2	2082.6 ± 1568.4	3078.8 ± 1158.1
hand-insert-v2	927.9 ± 1527.6	288.8 ± 932.1
drawer-close-v2	4824.5 ± 8.8	4825.3 ± 22.3
drawer-open-v2	3530.4 ± 1512.6	2218.8 ± 529.3
faucet-open-v2	3877.4 ± 598.1	4245.6 ± 575.2
faucet-close-v2	4167.1 ± 418.8	4624.1 ± 107.5
handle-press-side-v2	3423.3 ± 1075.6	3995.8 ± 1130.0
handle-press-v2	3397.9 ± 1752.1	3125.4 ± 1724.7
handle-pull-side-v2	3141.5 ± 1775.4	1474.2 ± 1392.9
handle-pull-v2	2845.7 ± 1970.6	2856.8 ± 1699.8
lever-pull-v2	3383.5 ± 1299.2	3125.6 ± 1433.1
peg-insert-side-v2	915.1 ± 1624.8	497.8 ± 1015.3
pick-place-wall-v2	649.8 ± 1288.6	288.2 ± 988.1
pick-out-of-hole-v2	1792.8 ± 1959.9	700.5 ± 1190.3
reach-v2	4144.2 ± 645.4	966.4 ± 842.9
push-back-v2	97.2 ± 611.3	566.3 ± 1042.9
push-v2	142.0 ± 335.7	64.6 ± 98.2
pick-place-v2	166.0 ± 759.2	7.3 ± 4.8
plate-slide-v2	4096.4 ± 1202.5	4306.3 ± 495.8
plate-slide-side-v2	2910.3 ± 616.1	2989.0 ± 1044.1
plate-slide-back-v2	4378.5 ± 373.0	3963.4 ± 927.3
plate-slide-back-side-v2	3872.0 ± 1151.6	4186.5 ± 772.6
soccer-v2	443.5 ± 785.5	480.4 ± 994.8
push-wall-v2	873.9 ± 1718.3	705.2 ± 1535.7
shelf-place-v2	204.5 ± 714.3	366.1 ± 919.0
sweep-into-v2	1297.9 ± 1661.5	506.2 ± 1368.5
sweep-v2	1397.1 ± 1922.3	599.1 ± 1359.2
window-open-v2	1453.6 ± 1101.4	3095.5 ± 751.1
window-close-v2	2963.9 ± 875.3	3177.6 ± 737.8
assembly-v2	2470.7 ± 1758.6	663.7 ± 28.7
button-press-topdown-wall-v2	1270.9 ± 214.5	1199.9 ± 203.7
hammer-v2	868.7 ± 983.4	1024.4 ± 1024.3
peg-unplug-side-v2	1439.1 ± 1817.8	136.4 ± 508.6
reach-wall-v2	4249.9 ± 536.0	4191.4 ± 412.9
stick-push-v2	1288.5 ± 1587.3	790.1 ± 1097.6
stick-pull-v2	601.4 ± 1346.3	287.1 ± 922.8
box-close-v2	2683.7 ± 1823.4	2273.3 ± 1707.3