Appendix

Model Card of DualMind

Model Details	
Model Type	Encoder-Decoder Transformer (Enc-Dec Control Transformer), built upon a ViT encoder [12], a TokenLearner [34], and a Control Transformer [36].
Training Process	The training process is divided into two phases. In Phase I, the entire Enc-Dec Control Transformer is trained with a self-supervised training objective. In Phase II,a small part of the model is trained using imitation learning with prompt conditions. The detailed training objective is described in Sec. 4.2.
Model Version	Initial release.
Intended Uses	
Primary Intended Uses	The proposed model aims to perform a wide range of control tasks spanning multiple domains, visual scenes and embodiments. Our intention is to create a general-purpose decision-making solution capable of handling various tasks using a single set of weights, without requiring task-specific fine-tuning.
Factors	
Relevant Factors	Multiple factors can influence the performance of the model. First, the quality of training dataset has influence on the results, including task diversity, behavior policy performance, data volume, etc. Second, model implementation hyperparameter setting, and training objectives will also alter the final performance.
Evaluation Factors	We report the performance of the model in multiple sets of tasks, and conducted ablation study in Sec. 5.3.2.
Metrics	
Model Performance Measures	Our downstream task performance is measured using success rate, SPL, and expert score, as detailed in Sec. A.5. The expert score is calculated in the same manner as GATO [32], while using a different dataset.
Decision thresholds	N/A
Approaches to Uncertainty and Variability	The model evaluation process inevitably involves uncertainties. In order to reduce the variance introduced during the evaluation, we employed 3 random seeds for the Habitat evaluation and 10 random seeds for the Metaworld evaluation.
Evaluation Data	
Datasets	Our DualMind is evaluated on multiple control tasks from Habitat and Metaworld. Both in-distribution and out-of-distribution tasks are considered. **Habitat:** Our experiments are focused on the ImageNav task, we chose 4 scenes . We hold out 5 Gibson scenes for the experiments of out-of-distribution tasks, as detailed in Sec. A.5. **Metaworld:** We select 45 training tasks on ML45 for the experiments of evaluation, and hold out 5 test tasks for the experiments of out-of-distribution, as detailed in Sec. A.5.
Motivation	Our evaluation of DualMind consists of two components. First, we evaluated its performance on in-distribution tasks to understand how well it handles tasks across domains, scenes, and embodiments using a single set of model weights. Second, we evaluated DualMind on out-of-distribution tasks to assess its ability to adapt to entirely new tasks.
Preprocessing	Observations are tokenized into the same embedding sequence before being input to transformer decoder, as detailed in Sec. 4.1.
Training Data	
Datasets	The model is trained using 100K episodes collected from Habitat and Metaworld, with 50k episodes ($\sim$3.26M interaction steps) on Habitat and 50K episodes ($\sim$3.82M interaction steps) on Metaworld, respectively.

Motivation	In order to ensure that DualMind can handle tasks across domains, scenes, and embodiments, we collected data for all tasks in Metaworld and all scenes in Habitat. The data collection process is detailed in Sec. A.3.
Preprocessing	The multi-domain data is tokenized into the same embedding sequence before being fed to the transformer decoder, as detailed in Sec. 4.1.
Quantitative Analyses	
Unitary Results	We evaluated the performance of DualMind on the Metaworld and Habitat benchmarks. In Sections 5.2 and 5.3.1, we demonstrate the general capabilities of DualMind across both Metaworld and Habitat tasks. Additionally, in Section 5.3.2, we analyze its performance on out-of-distribution tasks.
Ethical Considerations	
Data	Our data is collected from simulators of navigation and manipulation, and thus it does not include any unethical data.
Risks and Harms	Our current training and evaluation are conducted in simulators, and do not involve physical robots where model malfunctioning could lead to safety issues.
Mitigations	N/A
Caveats and Recommendation	
Future work	Our future work includes expanding DualMind to more domains and tasks, finding efficient solutions for handling longer context lengths in demonstrations, and enabling practical training in online interactive scenarios.

Table 3: Model card of DualMind, following the framework proposed by [23].

A. Implementation details

A.1. Model and hyperparameters

In this section, we provide a summary of the architecture and hyperparameters used in the Encoder-Decoder Control Transformer. Our model consists of a ViT encoder, a TokenLearner, and a Control Transformer. The Control Transformer we use is composed of 8 causal attention layers with 8 attention heads, 8 cross-attention layers with 8 attention heads, and an embedding dimension of 512. The ViT encoder is ViT-B/16 and we load pretrained weights from MultiMAE [3]. Instead of using mean pooling and a linear projection layer, we employ a TokenLearner that subsamples the 196 patch tokens output by the ViT encoder to 8 tokens, which are then passed to the Transformer decoder layers.

For both Phase I and Phase II, we utilize the default AdamW optimizer [20]. For Phase I, the learning rate and batch size are set to 5e-5 and 16, respectively, while for Phase II, they are set to 1e-4 and 128, respectively. Additionally, a context length of 6 is used in all models for both training and execution. Phase I has 175M trainable parameters while Phase II has 51.1M trainable parameters. All models are trained for 10 epochs in Phase I, and 10 epochs for Phase II, with additional training details provided in Sec. A.4.

A.2. Baselines architecture

We summary the differences between DualMind and Baselines in Table 4. The details of Baselines are listed below.

A.2.1 `IL-only`, `SMART-only` and `Jointly`.

The models `IL-only`, `SMART-only`, and `Jointly` all employ the same architecture as DualMind during Phase II. This architecture encompasses a Transformer encoder (with State and Action tokenizers), a decoder, and a XAtten. layer. The only difference between them is the modification of the training objectives and phase.

The `IL-only` model focuses solely on prompt-conditioned imitation learning during its training process. In contrast, the `SMART-only` model leverages SMART training objectives in a purely self-supervised learning context with prompt-conditioning. The `Jointly` model synthesizes these methods, employing both SMART objectives and prompt-conditioned imitation learning loss in its comprehensive training strategy.

	Training objectives	Model structure	Dual-phase
DualMind	Phase I: Self-superv. Phase II: IL-prompt	Phase I.:Enc-Dec Control Transformer Phase II: +XAtten.	✓
`IL-only`	IL-prompt	Enc-Dec Control Transformer +XAtten.	✗
`SMART-only`	Self-superv. prompt	Enc-Dec Control Transformer +XAtten.	✗
`Jointly`	Self-superv. + IL-prompt	Enc-Dec Control Transformer +XAtten.	✗
`GATO-CT`	IL-prompt	Enc-Dec Control Transformer	✗
`GATO*`	IL-prompt	GATO [32]	✗

Table 4: Comparisons of different baselines.

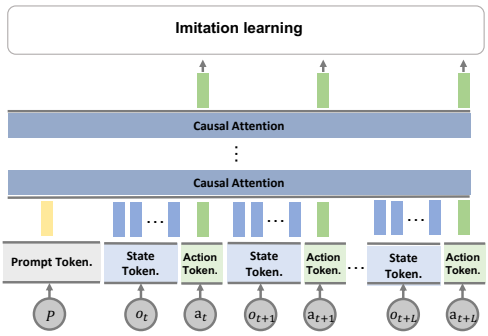

Figure 11: The architecture diagram of `GATO-CT`.

A.2.2 `GATO-CT` and `GATO*`

`GATO*`: GATO [32] (In this paper, we use **`GATO*`** to denote) is decode-only model, which imitates expert demonstrations from a vast dataset by prompting the model with the state and action subsequence. This model has 1.18 billion parameters and was trained on massive datasets, including 94.6k episodes from Metaworld. We include its reported performance on the Metaworld benchmark for reference.

`GATO-CT`: For a fair comparison, we used the same base model architecture (Enc-Dec Control Transformer), but replaced our XAtten.-based prompting approach with their proposed prefix prompting approach, denote as **`GATO-CT`**. Similar to `IL-only`, only imitation learning loss is used to predict future actions, But replace the XAtten. module with one that prefixes the model with prompt token. Details are provided in Fig. 11.

A.3. Data collection

Habitat. We collect shortest path episodes sampled from each of the 72 Gibson [39], 61 mp3d [8] and 800 hm3d [31] training scenes. These demonstrations are generated by greedily fitting actions to follow the geodesic shortest path to the nearest navigable goal object viewpoint. We hold out 5 Gibson scenes (hominy, Goffs, Hillsdale, Micanopy, and Rosser) for the experiments of out-of-distribution. The data we collected included RGB images ($3 \times 224 \times 224$), goal, and actions. We collected about 1000 episodes for each scene. Then, we divided the dataset and created the following dataset based on their intended purposes.

- **Habitat 50k.** We select all scenes of the Habitat dataset, and randomly sample about 50 episodes pre scene from the Habitat dataset. This data has 50K episodes and about ∼3.26M interaction steps. The dataset is used in Phase I and Phase II training.
- **Habitat 10k.** We randomly select 10 scenes of Habitat scenes, and randomly sample 1000 episodes pre scene from the Habitat dataset. This data has 10K episodes and about ∼0.54M interaction steps. The dataset is used to train the model in Phase II of the ablation study.
- **Out-of-distribution tasks.** We select 5 Gibson scenes (”Goffs”, ”Hominy”, ”Hillsdale”, ”Micanopy”, and ”Rosser”) held out, and randomly sample 10, 100, and 1000 episodes pre scenesfrom the Habitat dataset.

Metaworld. We collected data for all tasks in the MT50 [40] using scripted policies, which allowed us to generate expert demonstrations across an unlimited number of environment seeds. The data we collected included RGB images (3×224×224) rendered by the physical simulator, physics engine states, and actions. We collected 2000 episodes for each tasks. We use 45 tasks in the ML45 for Phase I, and hold out other 5 tasks (hand-insert-v2, door-lock-v2, door-unlock-v2, box-close-v2 and bin-picking-v2) for the experiments of out-of-distribution. Then, we divided the dataset and created the following dataset based on their intended purposes.

- **ML45.** We select 45 training tasks of ML45 in Metaworld, and randomly sample 1000 episodes pre task from the Metaworld dataset. This data has 45K episodes and about ∼3.40M interaction steps. The dataset is used in Phase I and Phase II training.
- **ML10.** We select 10 training tasks of ML10 in Metaworld, and randomly sample 1000 episodes pre task from the Metaworld dataset. This data has 10K episodes and about ∼0.79M interaction steps. The dataset is used to train the model in Phase II of the ablation study.
- **Out-of-distribution tasks.** We select 5 test tasks of ML45 in Metaworld (”hand-insert-v2”, ”door-unlock-v2”, ”door-lock-v2”, ”box-close-v2”, and ”bin-picking-v2”), and randomly sample 10, 100, and 1000 episodes pre task from the Metaworld dataset.

A.4. Training detail

Phase I. In Phase I, the entire model, except for the cross-attention layers (XAtten.), is trained using a self-supervised training objective on the ML45 dataset.

Phase II. In Phase II, we freeze the model encoder and only finetune a small part of the model, namely the Control Transformer, using imitation learning based on prompts. To encode the prompts, we use the CLIP encoder (CLIP/ViT-B/16) [27] and denote the resulting prompt sequence as P. The output sequence from each cross-attention layer is computed by $\text{softmax}(\frac{q_H k_p^T}{\sqrt{d}})v_P$, where H is the sequence of episodes and d is the embedding dimension. In ablation study, we use Habitat 10K and ML10 datasets for Phase II training dataset, while for the other experiments we use the Habitat 50K and ML45 datasets as training data for Phase II.

Out-of-distribution tasks. In Sec. 5.3.2, we use `DualMind`, `IL-only`, and `Scratch` for out-of-distribution tasks. `DualMind` and `IL-only` model are trained beforehand and further finetuned with few-shot demonstrations. `Scratch` refers to the model that is trained on few-shot demonstrations from randomly initialized model weights. We randomly select 10, 100 and 1000 episodes for few-shot learning. We use batch size bs = 64 and lr = 1e-4. We train all models for 10000 gradient steps. The data for the out-of-distribution tasks are generated in the same way as we did in Sec. A.3.

Ablation study. In Sec. 5.4, we use Phase I model pretrained on Habitat 50k and ML45 datasets. And the training parameters were the same as in Phase II except for the change in ablation condition and datasets.

A.5. Evaluation detail

Habitat. Habitat is an immersive navigation task that provides a visually realistic environment. Our experiments are focused on the ImageNav task, in which the agent navigates towards a target position based on a goal image. The agent should stop within 1000 steps and reach a distance of 1m from the target image. To conduct our evaluation, we chose 4 scenes (Convoy, Beach, Cooperstown and Eagerville). We hold out 5 Gibson scenes (hominy, Goffs, Hillsdale, Micanopy, and Rosser) for the experiments of out-of-distribution tasks. For each scene, we randomly select three difficulty levels based on path length (EASY: 1.5-3m, MEDIUM: 3-5m, and HARD: 5-10m), resulting in a total of 300 episodes per scene. The metrics of the Habitat benchmark are listed below:

- **Success Rate(SR) and Success weighted by Path Length(SPL).** The success rate(SR) and success eighted by Path Length(SPL), proposed by [2], are estimated over 100 episodes on 4 scenes with 3 difficulty levels per scene, for a total of 1200 episodes per seed.

Metaworld. Metaworld is a benchmark of 50 diverse simulated manipulation tasks. We select 45 training tasks on ML45 for the experiments of evaluation, and hold out 5 test tasks (”hand-insert-v2”, ”door-unlock-v2”, ”door-lock-v2”, ”box-close-v2”, and ”bin-picking-v2”) for the experiments of out-of-distribution. The metrics of the Metaworld benchmark are listed below.

- **Success Rate(SR)**. We refer to the evaluation method in Metaworld [40]. The success rate is estimated over 10 seeds per task.

- **Expert Score**. The expert score is a measure of the difference between the performance of agents and experts, and is calculated as the ratio of the return obtained by agents to the expert return. We use the same expert return calculation method as GATO [32].

$$\max_{j \in [0,1,\ldots,N-W]} \left(\sum_{i=j}^{j+L-1} \frac{R_i}{W} \right)$$

where N it the total number of collected episodes for the task, W is the window size, and R_i is the total return for episode i.

B. More experiments

B.1. Comparisons of varying context length

We conducted experiments on different context lengths, as illustrated in Fig. 12 and Fig. 13. On the navigation tasks in Habitat, long-range temporal dependencies are important for decision-making. As a result, the model's performance is improved progressively as the length of the context increases, as shown in Fig. 12. On the other hand, we observed that setting the context length to 6 leads to better performance on the Metaworld dataset, as demonstrated in Fig. 13. Therefore, we choose a context length of 6 as means of balancing performance and compute cost. However, if one seeks to capture long-term temporal dependence, increasing the context length may be necessary.

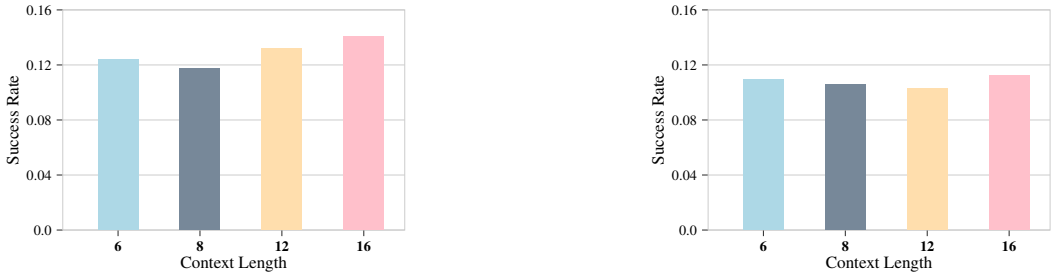

Figure 12: Comparison of varying **context length** on *Habitat*,and compare agents by Success Rate (SR) (left) and Success weighted by Path Length (SPL) (right).

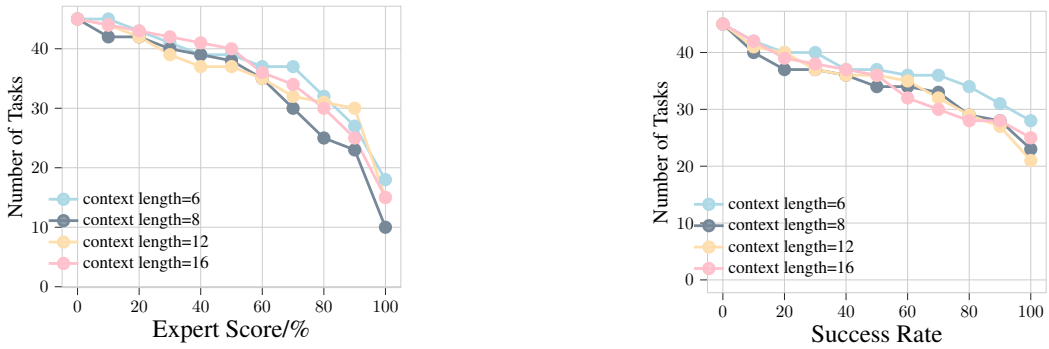

Figure 13: Comparisons of varying **context length** on *MetaWorld 45 tasks* on Percentage of Expert Score (PES) (left) and Success Rate (SR) (right).

B.2. Comparison with vision tokenization

We conducted a set of experiments to demonstrate the effectiveness of the multi-state tokens module (i.e., TokenLearner), by comparing it with a single-state tokens module that uses mean pooling and a linear projection layer to convert patch tokens to one token. In contrast to other ablation studies, we trained both Phase I and Phase II on the ML45 and Habitat 50k datasets.

The results in Table 5 indicate a minor difference between the two on habitat, but an average success rate difference of approximately 0.09 on ML45. These findings support our expectation that multi-state tokens can extract additional information from the encoder to improve decision-making and enhance overall learning performance.

tokenization	Habitat	Metaworld
multi-state token.	0.1239	0.802
single-state token.	0.1217	0.713

Table 5: Comparisons of tokenization methods on *Habitat* and *MetaWorld 45* tasks, measured by Success Rate (SR).

B.3. Prompt conditioning discussion

Implementation details. In Sec. 5.4, we conducted an ablation study by comparing two prompt conditioning approaches: prefix and XAtten. prompting. The prefix approach is a conventional prompting method that splices the prompt sequences in front of the token sequences, which are directly fed into the Transformer decoder layers. In contrast, XAtten. prompting uses a cross-attention layer to fuse the prompt sequences and token sequences together. We utilized the base model that was pretrained on Habitat 50 and ML45 after Phase I. We use Habitat 10K and ML10 datasets for Phase II training dataset.

Discussion. In the experiments discussed in Sec.5.4, it was found that XAtten. prompting outperforms prefix prompting. This suggests that the cross attention mechanism is effective in establishing a strong connection between prompt and token sequences, which has also been demonstrated in other recent works, such as Vima[17] and Stable Diffusion [33]. One potential limitation of prefix prompting is that the prompt token sequence may be too short to attract sufficient attention from the attention mechanism, leading to suboptimal performance. To address this, future research may explore alternative encoding methods for prompts that can better capture the information necessary for guiding the model's output.

B.4. Attention visualization

In Figure 14, we provide additional attention maps that reveal how DualMind tends to focus on the object being manipulated, as well as its surrounding context and relevant visual cues (such as "plate-slide-v2", "push-v2", and "hammer-v2") when performing manipulation tasks in MetaWorld. Furthermore, the attention maps show that the model focuses on the location of the item being manipulated, and then interacts with the corresponding item to complete the task. In Habitat, our model (DualMind) focuses more on exploring the scene and then locating the goal, as illustrated in Figure 14. The attention maps demonstrate that DualMind quickly identified the location of the goal image at the outset. Despite that there are obstacles blocking the shortest path, DualMind was able to eventually reach the goal.

B.5. Performance on each tasks

We show the detailed results of all models on Metaworld and Habitat in Table 6, Table 7, and Table 8. Specifically, Table 6 presents the Habitat results, while Table 7 and Table 8 present the Metaworld results.

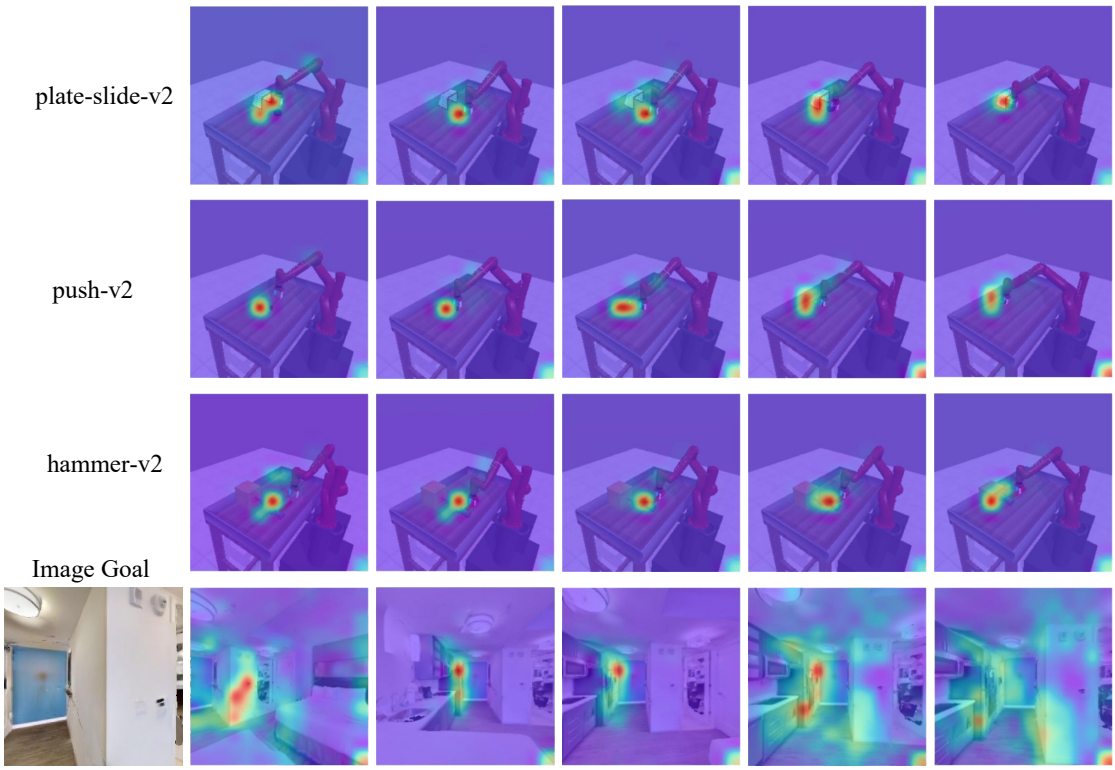

Figure 14: More attention map visualization. On Metaworld, the attention maps show that the model focuses on the location of the item being manipulated, and then interacts with the corresponding item to complete the task. On Habitat, DualMind focuses more on exploring the scene and then locating the goal.

	DualMind		DualMind/single	
scene	SR	SPL	SR	SPL
Convoy(easy)	0.143± 0.021	0.124±0.017	0.160±0.026	0.115±0.027
Convoy(medium)	0.150±0.017	0.144±0.020	0.157±0.006	0.126±0.009
Convoy(hard)	0.067±0.025	0.062±0.029	0.173±0.042	0.162±0.039
Beach(easy)	0.170±0.044	0.144±0.045	0.170±0.053	0.133±0.036
Beach(medium)	0.157±0.015	0.146±0.012	0.200±0.050	0.165±0.050
Beach(hard)	0.133±0.015	0.128±0.014	0.227±0.045	0.215±0.047
Cooperstown(easy)	0.147±0.021	0.122±0.030	0.180±0.070	0.141±0.058
Cooperstown(medium)	0.110±0.035	0.103±0.027	0.190±0.046	0.175±0.044
Cooperstown(hard)	0.073±0.021	0.070±0.020	0.140±0.046	0.132±0.043
Eagerville(easy)	0.113±0.025	0.079±0.027	0.087±0.015	0.054±0.003
Eagerville(medium)	0.127±0.029	0.106±0.020	0.107±0.025	0.087±0.019
Eagerville(hard)	0.097±0.046	0.081±0.043	0.147±0.021	0.130±0.018
	Jointly		Jointly/single	
scene	SR	SPL	SR	SPL
Convoy(easy)	0.130±0.020	0.122±0.023	0.173±0.031	0.120±0.020
Convoy(medium)	0.080±0.000	0.076±0.002	0.143±0.031	0.111±0.035
Convoy(hard)	0.050±0.010	0.049±0.011	0.177±0.015	0.160±0.017
Beach(easy)	0.137±0.021	0.115±0.013	0.220±0.056	0.156±0.026
Beach(medium)	0.093±0.021	0.086±0.019	0.230±0.030	0.172±0.015
Beach(hard)	0.057±0.021	0.054±0.021	0.147±0.042	0.115±0.018
Cooperstown(easy)	0.117±0.035	0.106±0.034	0.173±0.021	0.139±0.027
Cooperstown(medium)	0.077±0.006	0.069±0.007	0.240±0.026	0.219±0.027
Cooperstown(hard)	0.080±0.020	0.076±0.022	0.183±0.032	0.167±0.032
Eagerville(easy)	0.137±0.035	0.116±0.035	0.100±0.010	0.058±0.002
Eagerville(medium)	0.097±0.015	0.085±0.016	0.180±0.030	0.112±0.021
Eagerville(hard)	0.067±0.023	0.061±0.022	0.193±0.040	0.150±0.020
	IL-only		IL-only/single	
scene	SR	SPL	SR	SPL
Convoy(easy)	0.113±0.006	0.110±0.006	0.110±0.010	0.09±0.011
Convoy(medium)	0.040±0.026	0.039±0.026	0.037±0.025	0.030±0.021
Convoy(hard)	0.027±0.006	0.027±0.006	0.053±0.012	0.043±0.011
Beach(easy)	0.103±0.055	0.095±0.051	0.070±0.010	0.052±0.010
Beach(medium)	0.053±0.035	0.050±0.033	0.050±0.010	0.035±0.0135
Beach(hard)	0.030±0.017	0.027±0.015	0.033±0.012	0.027±0.010
Cooperstown(easy)	0.080±0.010	0.074±0.010	0.103±0.015	0.076±0.01
Cooperstown(medium)	0.043±0.012	0.041±0.023	0.053±0.012	0.038±0.013
Cooperstown(hard)	0.047±0.020	0.045±0.009	0.050±0.000	0.039±0.006
Eagerville(easy)	0.067±0.025	0.064±0.025	0.063±0.021	0.042±0.023
Eagerville(medium)	0.070±0.030	0.054±0.018	0.033±0.021	0.022 ±0.01
Eagerville(hard)	0.047±0.021	0.040±0.024	0.033±0.021	0.026±0.014
	SMART-only		SMART-only/single	
scene	SR	SPL	SR	SPL
Convoy(easy)	0.113±0.006	0.088±0.003	0.007±0.006	0.007±0.006
Convoy(medium)	0.003±0.006	0.003±0.005	0.0±0.0	0.0±0.0
Convoy(hard)	0.007±0.006	0.005±0.004	0.0±0.0	0.0±0.0
Beach(easy)	0.063±0.025	0.044±0.014	0.007±0.012	0.007±0.012
Beach(medium)	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Beach(hard)	0.010±0.010	0.008±0.009	0.0±0.0	0.0±0.0
Cooperstown(easy)	0.090±0.020	0.079±0.018	0.013±0.006	0.013±0.006
Cooperstown(medium)	0.010±0.010	0.007±0.006	0.0±0.0	0.0±0.0
Cooperstown(hard)	0.003±0.006	0.003±0.006	0.0±0.0	0.0±0.0
Eagerville(easy)	0.087±0.006	0.076±0.003	0.017±0.006	0.016±0.005
Eagerville(medium)	0.023±0.006	0.017±0.004	0.0±0.0	0.0±0.0
Eagerville(hard)	0.010±0.010	0.008±0.008	0.0±0.0	0.0±0.0

Table 6: performance on each tasks on Habitat

task	DualMind		DualMind/single		Jointly		Jointly/single	
	SR	return	SR	return	SR	return	SR	return
assembly-v2	0.9	1039.1	1.0	1276.0	0.0	198.8	0.0	477.0
basketball-v2	0.3	400.0	0.5	621.7	0.0	10.6	0.0	42.4
button-press-topdown-v2	1.0	364.9	1.0	1175.4	0.6	113.9	0.6	45.2
button-press-topdown-wall-v2)	1.0	0.0	1.0	0.0	0.7	55.2	1.0	39.5
button-press-v2	1.0	347.3	1.0	357.6	0.7	318.2	0.5	326.2
button-press-wall-v2	0.3	1373.7	1.0	1195.2	0.0	128.8	0.0	0.5
coffee-button-v2	1.0	301.0	1.0	299.2	1.0	304.3	0.6	268.6
coffee-pull-v2	1.0	429.5	1.0	407.3	0.8	303.8	0.9	324.2
coffee-push-v2	1.0	443.6	1.0	509.0	0.0	30.3	0.2	66.9
dial-turn-v2	1.0	1220.0	1.0	1203.3	0.0	61.6	0.3	17.7
disassemble-v2	1.0	615.3	1.0	553.4	0.0	220.9	0.0	213.2
door-close-v2	1.0	946.0	1.0	754.9	0.1	2956.1	1.0	1286.4
door-open-v2	1.0	1756.7	1.0	1775.4	0.1	696.5	0.9	1457.9
drawer-close-v2	1.0	61.4	1.0	81.2	0.1	4.9	0.4	24.6
drawer-open-v2	1.0	1965.0	1.0	1989.5	0.7	1517.8	0.5	1103.6
faucet-open-v2	0.3	1693.4	1.0	2192.2	0.0	1388.9	0.0	1276.4
faucet-close-v2	0.1	1624.0	0.0	1695.1	0.0	1856.1	0.2	2075.1
hammer-v2	1.0	951.9	1.0	929.5	0.0	675.8	0.1	869.2
handle-press-side-v2	1.0	839.9	1.0	808.8	0.7	307.2	1.0	877.9
handle-press-v2	1.0	671.1	1.0	782.7	0.4	195.6	0.7	427.7
handle-pull-side-v2	0.8	580.4	0.0	29.1	0.0	12.4	0.0	11.4
handle-pull-v2	0.7	182.9	0.2	177.5	0.1	70.4	0.5	139.9
lever-pull-v2	0.0	291.1	0.8	946.7	0.1	420.3	0.0	283.8
peg-insert-side-v2	0.9	990.8	0.7	909.5	0.5	1413.7	0.3	1096.3
pick-place-wall-v2	1.0	656.9	1.0	1698.2	0.0	0.1	0.0	14.0
pick-out-of-hole-v2	0.0	261.1	0.0	362.4	0.0	25.9	0.0	12.3
reach-v2	0.1	2507.2	0.0	2374.8	0.0	241.8	0.0	598.6
push-back-v2	1.0	193.7	1.0	283.7	0.0	6.5	0.0	6.4
push-v2	1.0	1264.1	1.0	1446.0	0.0	22.7	0.0	23.4
pick-place-v2	0.7	608.4	1.0	303.0	0.0	7.2	0.0	19.2
plate-slide-v2	1.0	1255.4	1.0	1214.2	0.0	269.7	0.2	417.6
plate-slide-side-v2	1.0	1281.6	1.0	1278.6	0.0	143.1	0.2	397.8
plate-slide-back-v2	1.0	1207.5	1.0	1170.6	0.8	909.0	0.4	618.6
plate-slide-back-side-v2	1.0	1340.7	1.0	1321.9	0.1	499.6	0.6	782.1
peg-unplug-side-v2	0.8	343.9	1.0	344.0	0.1	109.2	0.3	122.0
soccer-v2	0.0	336.4	0.0	329.6	0.1	180.3	0.0	164.3
stick-push-v2	1.0	1328.5	1.0	1316.9	0.0	24.2	0.5	475.5
stick-pull-v2	0.9	190.4	1.0	648.5	0.0	6.3	0.1	144.8
push-wall-v2	1.0	1387.4	1.0	1782.3	0.0	51.3	0.0	49.4
reach-wall-v2	0.5	3151.8	0.0	4190.0	0.0	341.4	0.0	714.7
shelf-place-v2	0.8	752.2	0.9	864.4	0.0	0.0	0.0	0.1
sweep-into-v2	1.0	880.6	0.8	783.1	0.0	47.5	0.0	52.0
sweep-v2	1.0	1346.6	1.0	1108.6	0.0	93.3	0.0	91.8
window-open-v2	1.0	438.6	1.0	494.6	0.5	379.8	0.2	436.8
window-close-v2	1.0	784.2	1.0	806.3	0.5	799.5	0.4	541.1

Table 7: The detailed Metaworld ML45 results of the DualMind, DualMind/single, Jointly and Jointly/single on each tasks.

	IL-only		IL-only/single		SMART-only		SMART-only/single	
	SR	return	SR	return	SR	return	SR	return
assembly-v2	0.0	252.1	0.0	169.1	0.0	197.2	0.0	189.0
basketball-v2	0.0	13.9	0.0	5.1	0.0	2.0	0.0	1.3
button-press-topdown-v2	0.0	194.3	0.0	33.3	0.0	116.4	0.0	0.1
button-press-topdown-wall-v2	0.6	88.6	0.0	1.3	0.0	35.7	0.0	14.5
button-press-v2	0.3	366.0	0.0	43.4	0.0	53.3	0.0	45.3
button-press-wall-v2	0.0	75.9	0.0	19.2	0.0	59.1	0.0	25.2
coffee-button-v2	1.0	301.0	0.0	38.6	0.6	297.3	0.0	65.1
coffee-pull-v2	0.9	365.8	0.0	13.5	0.0	11.6	0.0	11.9
coffee-push-v2	0.0	83.6	0.0	13.1	0.0	10.9	0.0	5.1
dial-turn-v2	0.0	19.3	0.0	6.7	0.0	4.4	0.0	8.3
disassemble-v2	0.0	210.9	0.0	206.6	0.0	204.8	0.0	206.1
door-close-v2	0.2	2742.7	0.0	30.4	0.0	659.8	0.2	327.4
door-open-v2	0.4	1169.6	0.0	169.6	0.0	212.3	0.0	383.7
drawer-close-v2	0.0	0.0	1.0	71.3	0.0	2.3	0.0	0.0
drawer-open-v2	1.0	1976.5	0.0	389.6	0.0	493.4	0.0	389.9
faucet-open-v2	0.1	1547.4	0.0	427.1	0.0	490.4	0.0	302.0
faucet-close-v2	0.0	1062.5	0.0	453.2	0.0	863.6	0.0	552.3
hammer-v2	0.0	588.5	0.0	528.0	0.0	263.3	0.0	585.6
handle-press-side-v2	0.6	235.4	0.8	493.9	0.0	26.3	0.0	28.7
handle-press-v2	0.0	86.1	0.0	18.5	0.0	34.3	0.0	22.9
handle-pull-side-v2	0.3	12.1	0.0	10.7	0.0	2.1	0.0	2.5
handle-pull-v2	0.2	82.5	0.0	6.4	0.0	14.4	0.0	4.4
lever-pull-v2	0.0	350.2	0.0	24.2	0.0	125.0	0.0	90.0
peg-insert-side-v2	0.9	1289.4	0.0	2.2	0.0	2.4	0.0	1.7
pick-place-wall-v2	0.0	27.4	0.0	0.0	0.0	0.0	0.0	0.0
pick-out-of-hole-v2	0.0	19.1	0.0	3.4	0.0	3.1	0.0	1.2
reach-v2	0.0	195.0	0.0	122.3	0.0	144.5	0.0	195.3
push-back-v2	0.0	5.0	0.0	1.7	0.0	2.3	0.0	1.7
push-v2	0.0	21.8	0.0	10.2	0.0	3.8	0.0	4.6
pick-place-v2	0.0	6.7	0.0	3.0	0.0	2.3	0.0	3.2
plate-slide-v2	0.3	393.9	0.0	72.1	0.0	97.2	0.0	44.2
plate-slide-side-v2	0.0	45.8	0.2	419.0	0.0	20.6	0.0	5.0
plate-slide-back-v2	0.7	1027.7	0.0	43.3	0.0	48.3	0.0	21.8
plate-slide-back-side-v2	0.0	200.1	0.0	1185.7	0.0	25.5	0.0	25.6
peg-unplug-side-v2	0.2	131.4	0.0	3.6	0.0	2.7	0.0	2.8
soccer-v2	0.0	21.0	0.0	38.1	0.0	3.4	0.0	6.9
stick-push-v2	0.0	16.7	0.0	5.7	0.0	1.9	0.0	3.1
stick-pull-v2	0.0	6.6	0.0	5.8	0.0	2.2	0.0	6.9
push-wall-v2	0.0	24.4	0.0	18.0	0.0	3.9	0.0	5.0
reach-wall-v2	0.0	558.7	0.0	305.2	0.0	159.9	0.0	435.3
shelf-place-v2	0.0	214.2	0.0	0.0	0.0	0.0	0.0	0.0
sweep-into-v2	0.0	55.3	0.0	8.3	0.0	10.7	0.0	9.1
sweep-v2	0.0	83.6	0.0	15.9	0.0	17.1	0.0	13.7
window-open-v2	1.0	449.5	0.0	101.4	0.0	91.4	0.0	92.7
window-close-v2	0.1	462.9	0.0	10.9	0.0	374.4	0.0	216.0

Table 8: The detailed Metaworld ML45 results of the IL-only, IL-only/single, SMART-only and SMART-only/single on each tasks.