Task-conditioned adaptation of visual features in multi-task policy learning – Supplementary Material

Pierre Marza [1] Laetitia Matignon[2] Olivier Simonin[1] Christian Wolf[3]
[1]INSA Lyon [2]UCBL [3]Naver Labs Europe
{pierre.marza, olivier.simonin}@insa-lyon.fr
laetitia.matignon@univ-lyon1.fr, christian.wolf@naverlabs.com

Project Page: https://pierremarza.github.io/projects/task_conditioned_adaptation/

A. Validation performance curves

Figure 1 shows the evolution of the validation score as a function of training epochs for rows (b)-(f) in Table 1 of the main paper. These curves showcase the gain brought by both middle and top adapters, and the positive impact of conditioning them on the task at hand.

B. Additional ablation studies

Impact of conditioning the policy on the task at hand — Row (b) in Table 1 reaches the same performance, even slightly better, as row (a), showing that when adapters are conditioned on the task at hand, conditioning the policy itself is not necessary. This seems to indicate that conditioned adapters already insert task-related information into visual embeddings fed to the policy.

Using the 'CLS' token representation as input to the policy — Row (c) in Table 1 performs worse than row (a), indicating that our introduced tokens aggregation layer ψ improves over the strategy used in previous work consisting in feeding the output 'CLS' token to the policy. This confirms our assumption that the 'CLS' token is undertrained under the MAE pre-training task.

Impact of the middle adapters when using top adapters — Row (d) in Table 1 also performs worse compared with row (a), showing that when training a conditioned top adapter, middle adapters are still very important, bringing a significant boost in performance.

C. Impact on other visual backbones

Table 2 shows the impact of our task-conditioned adapters on the visual features extracted by two other SOTA ViT-B backbones, i.e. PVR [23] and MVP [27]. As can be seen, our adapters bring a boost in policy performance for both pre-trained backbones, generalizing the conclusions obtained with VC-1.

D. Visualizing the influence of task-conditioned adaptation

This section focuses on investigating the impact of the introduced adapters on the processing of visual features. All sequences of visual frames used in the following experiments are taken from a held-out set of expert trajectories not used at training time.

Influence of middle adapters on ViT attention maps — We visualize here the attention map of the last layer of the vision encoder. To this end, we sum attention maps for all tokens and all heads, and normalize them between 0 and 1. These visualizations are shown in Figures 2 and 3. The first row shows a sequence of visual frames and below, for each model variant (No Adapter, Middle Adapter (NC), Middle Adapter (C)), one can see the attention map overlaid on top of the visual frame and displayed below as a colored heatmap.

As can be seen in Figures 2 and 3, the middle adapters help focus the attention on the most important parts of the image compared with vanilla VC-1 attention without adapters. When adapters are not conditioned (NC), they tend to either produce very narrow (Figure 2 and first frames in Figure 3) or quite broad (last frames in Figure 3) attention. Conditioning on the task at hand keeps the focus on important regions and leads to covering the entire objects of interest and important agent parts. Most importantly, Figures 2 and 3 show that, when adapters are conditioned on the task embedding, more attention is put on the final goal in all frames, while this is not the case for unconditioned adapters.

Conditioning middle adapters helps insert task-related information into visual embeddings — In order to study the underlying mechanisms of adapter modules, we examine the content of produced visual embeddings. Figure 4 shows t-SNE plots of visual embeddings for a set of

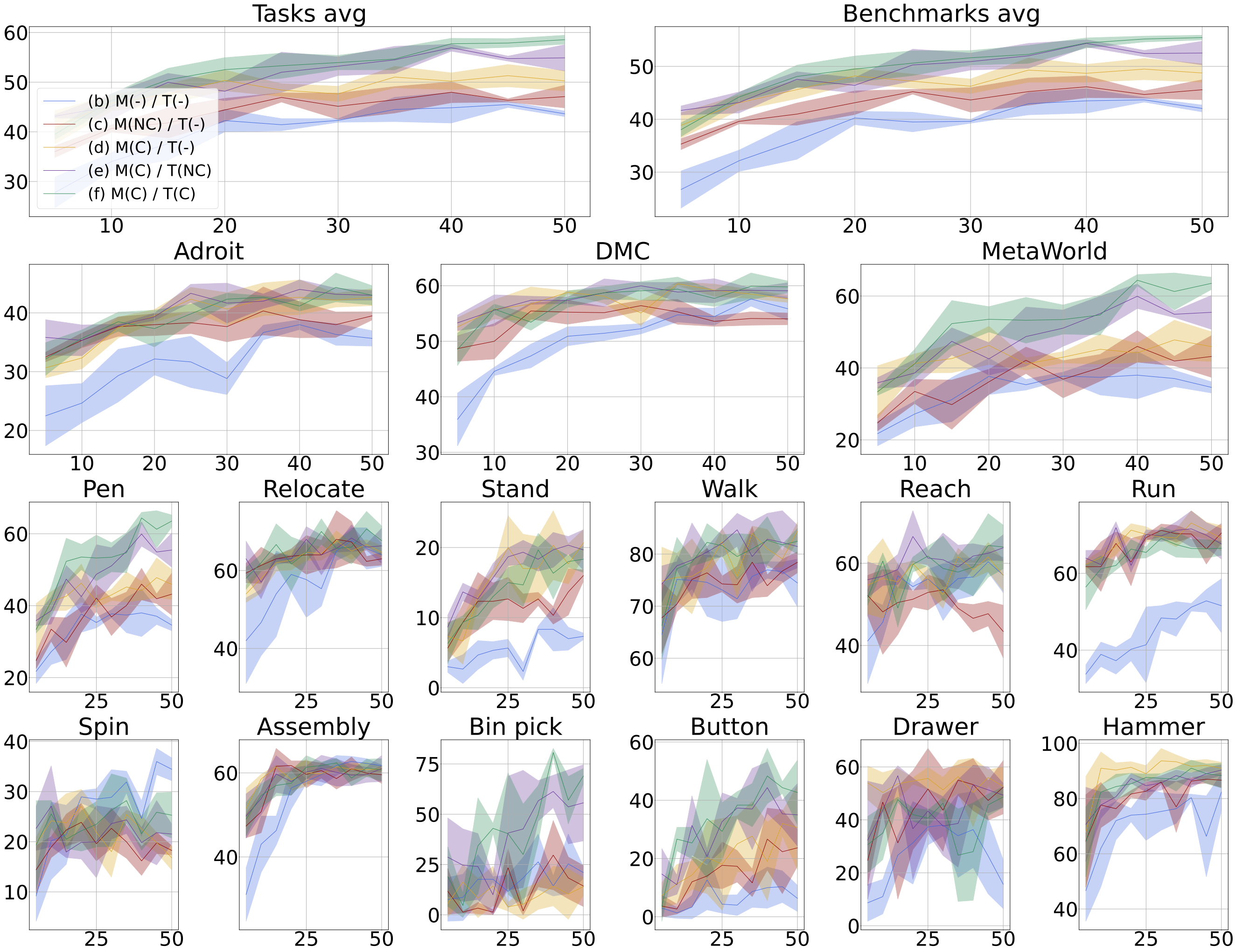

Figure 1. Known task — **Impact of visual adapters**: Evolution of the validation performance during training for rows (b)-(f) in Table 1 of the main paper. In the legend, M and T refer respectively to the state of middle and top adapters, and -, NC or C mean they are absent, not conditioned or conditioned on the task embedding. On all plots, the y-axis represents the performance score and the x-axis corresponds to the training epoch. Colored lines represent the evolution of mean performance over 3 training runs (3 random seeds) and shaded areas represent standard deviation.

frames for both the DMC *Stand* and *Walk* tasks. Visual observations are identical for both tasks at the beginning of rollouts, and very similar in the rest of the sequences, making it very hard to distinguish between these 2 tasks from a visual observation only. Embeddings from the conditioned middle adapters form two well-separated clusters, showcasing the task-related information brought by conditioning adapters on the task at hand.

E. Non-linear probing of actions

Table 3 shows the performance of a probing MLP network trained to regress the expert action to take from the visual embedding of a single frame only. As can be seen, its performance improves drastically when trained on embeddings predicted by a vision encoder composed of conditioned middle and top adapters. A conditioned top adapter thus inserts action-related information within visual embeddings.

F. Diversity of known tasks

Table 5 (c) and Table 4 (b) show a model trained on MetaWorld only, which performs better on MetaWorld than models trained on all 3 benchmarks (the domain gap between them is large). The lower performance on MetaWorld when training on all 3 benchmarks is largely outweighed by the ability to address Adroit and DMC.

Table 1. Known task — **Additional ablation studies**: Validation and test performance on known tasks of different neural variants. Row (a) is equivalent to row (f) in Table 1 of the main paper. When using conditioned adapters, giving the task embedding as input to the policy is not necessary. Our introduced tokens aggregation layer is better than using the representation of the 'CLS' token. Finally, when training a conditioned top adapter, middle adapters are still important and bring a boost in performance. **Cond** π: policy conditioned on the task embedding – C: Conditioned – **'CLS' token**: using the 'CLS' token representation as the frame embedding fed to the policy. Performance is reported as *mean* ± *std* over 3 training runs (seeds).

	Cond.	**Adapters**		**'CLS'**	**Multi-task performance**									
	π	Mid.	Top	**token**	Adroit		DMC		MetaWorld		Benchmarks avg		Tasks avg	
					Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
(a)	✓	C	C	–	42.0 ± 0.8	42.3 ± 1.0	59.9 ± 0.9	60.0 ± 0.5	65.3 ± 1.0	54.5 ± 3.3	55.8 ± 0.1	52.3 ± 1.0	59.2 ± 0.1	54.8 ± 1.2
(b)	–	C	C	–	42.3 ± 2.0	40.8 ± 3.0	59.2 ± 1.0	59.2 ± 2.3	68.7 ± 2.2	57.6 ± 3.4	56.8 ± 0.8	52.5 ± 0.9	60.4 ± 0.8	55.5 ± 0.5
(c)	✓	C	C	✓	38.8 ± 5.9	36.2 ± 2.3	57.8 ± 1.9	58.1 ± 2.6	57.5 ± 8.0	50.9 ± 4.3	51.4 ± 2.8	48.4 ± 0.6	54.5 ± 2.9	51.4 ± 1.3
(d)	✓	–	C	–	34.7 ± 1.5	34.7 ± 3.3	51.4 ± 0.4	52.1 ± 0.5	53.6 ± 4.4	44.5 ± 4.7	46.6 ± 1.6	43.8 ± 1.8	49.5 ± 2.0	46.0 ± 1.9

Table 2. Known task — **Impact on other visual backbones**: Validation and test performance on known tasks for two additional visual backbones (PVR [23] and MVP [27]). Our task-conditioned adapters improve the extracted visual features in both cases, leading to higher multi-task policy performance. Performance is reported as *mean* ± *std* over 3 training runs (seeds).

ViT	**Ours**	**Multi-task performance**									
		Adroit		DMC		MetaWorld		Benchmarks avg		Tasks avg	
		Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
PVR [23]	–	34.7±2.8	30.2± 1.0	58.1±3.0	55.3± 7.2	41.8± 1.7	33.5± 1.4	44.8± 1.3	39.6± 3.0	47.4± 1.3	42.0± 3.5
	✓	**43.3**± 2.5	**41.0**± 5.1	**61.9**± 1.3	**61.3**± 1.2	**66.8**± 2.3	**55.7**± 3.4	**57.3**± 1.0	**52.7**± 3.1	**60.9**± 0.8	**55.6**± 2.7
MVP [27]	–	38.0±1.3	34.3±2.4	56.5±2.1	56.5±1.7	42.8±6.9	35.9±5.6	45.8±1.5	42.2±2.2	47.7±1.9	44.2±2.2
	✓	**47.7**±5.9	**46.2**±2.4	57.3±2.9	56.9±2.5	**64.9**±12.1	**55.4**±12.2	**56.6**±4.0	**52.8**±2.6	**58.9**±4.4	**54.5**±3.8

Table 3. **Non-linear probing of actions**: we explore the performance of action regression from the visual embedding of a single frame. Considered metrics are the Mean Squared Error (MSE) and coefficient of determination (R^2). The top adapter seems to insert action-related information into the visual embedding, as the probing MLP achieves the best performance.

	Middle adapters	Top adapter	MSE	R^2
(a)	–	–	0.067	0.69
(b)	NC	–	0.069	0.57
(c)	C	–	0.069	0.59
(d)	C	NC	0.037	0.90
(e)	C	C	**0.034**	**0.92**

Table 4. Known task — **Diversity of known tasks**: Validation and test performance on MetaWorld known tasks when our approach with task-conditioned adapters is either trained on the three considered benchmarks (Adroit, DMC, MetaWorld), or on tasks from MetaWorld only. As expected, the model trained on known tasks from MetaWorld only reaches higher performance. Performance is reported as *mean* ± *std* over 3 training runs (seeds).

	Training	**MetaWorld**	
		Val	Test
(a)	All 3 benchmarks	65.3±1.0	54.5 ±3.3
(b)	MetaWorld only	75.6±1.6	67.8±2.6

G. Few-shot adaptation baseline

Table 5 compares model finetuning on new tasks (b) with our task embedding search (a). As expected, (b) performs better but task embedding search (a) solves a harder problem, as we keep a single policy. Our adapters can thus be used in 2 settings: (i) task embedding search, keeping a single policy addressing all tasks (low memory footprint), (ii) task-specific fine-tuning to reach the best performance possible if memory is not an issue (one specific set of 130M parameters for each task).

H. Architecture details

Vision encoder — we use a ViT-B backbone, initialized from VC-1 weights, as the base vision encoder ϕ. It is made of 12 self-attention layers, each composed of 12 attention heads, with a hidden size of 768. The input image of size 224×224×3 is divided into a grid of 14×14 patches, where each patch has thus a size of 16×16 pixels. An additional 'CLS' token is appended to the sequence of image tokens

Table 5. Few-shot — Performance of a finetuned baseline (*Ft.*) and task embedding search (*TE opt.*) for a policy either trained on MetaWorld only (*MV*) or all 3 benchmarks (*All 3*). t_i^u refers to the i-th unknown task. Performance is reported as *mean* ± *std* over 3 training runs (seeds).

	Opt.	**Train.**	**Setting**	t_0^u	t_1^u	t_2^u	t_3^u	t_4^u	t_5^u	t_6^u	t_7^u	t_8^u	t_9^u	t_{10}^u	t_{11}^u	t_{12}^u	t_{13}^u	t_{14}^u	**Mean**
(a)	TE opt.	All 3	Single policy	2 ± 4	1 ± 2	55 ± 26	41 ± 21	4 ± 4	34 ± 10	81 ± 21	72 ± 12	0 ± 0	11 ± 17	34 ± 30	48 ± 6	47 ± 2	53 ± 21	4 ± 1	33 ± 2
(b)	Ft.	All 3	15 policies	1 ± 2	0 ± 0	49 ± 44	69 ± 22	5 ± 2	62 ± 8	96 ± 4	91 ± 7	2 ± 3	54 ± 8	50 ± 4	22 ± 19	66 ± 7	89 ± 1	3 ± 2	44 ± 3
(c)	TE opt.	MW	Single policy	3 ± 6	0 ± 0	56 ± 44	64 ± 20	1 ± 2	69 ± 19	100 ± 0	77 ± 20	0 ± 1	6 ± 6	22 ± 38	19 ± 3	55 ± 13	57 ± 17	5 ± 3	36 ± 5

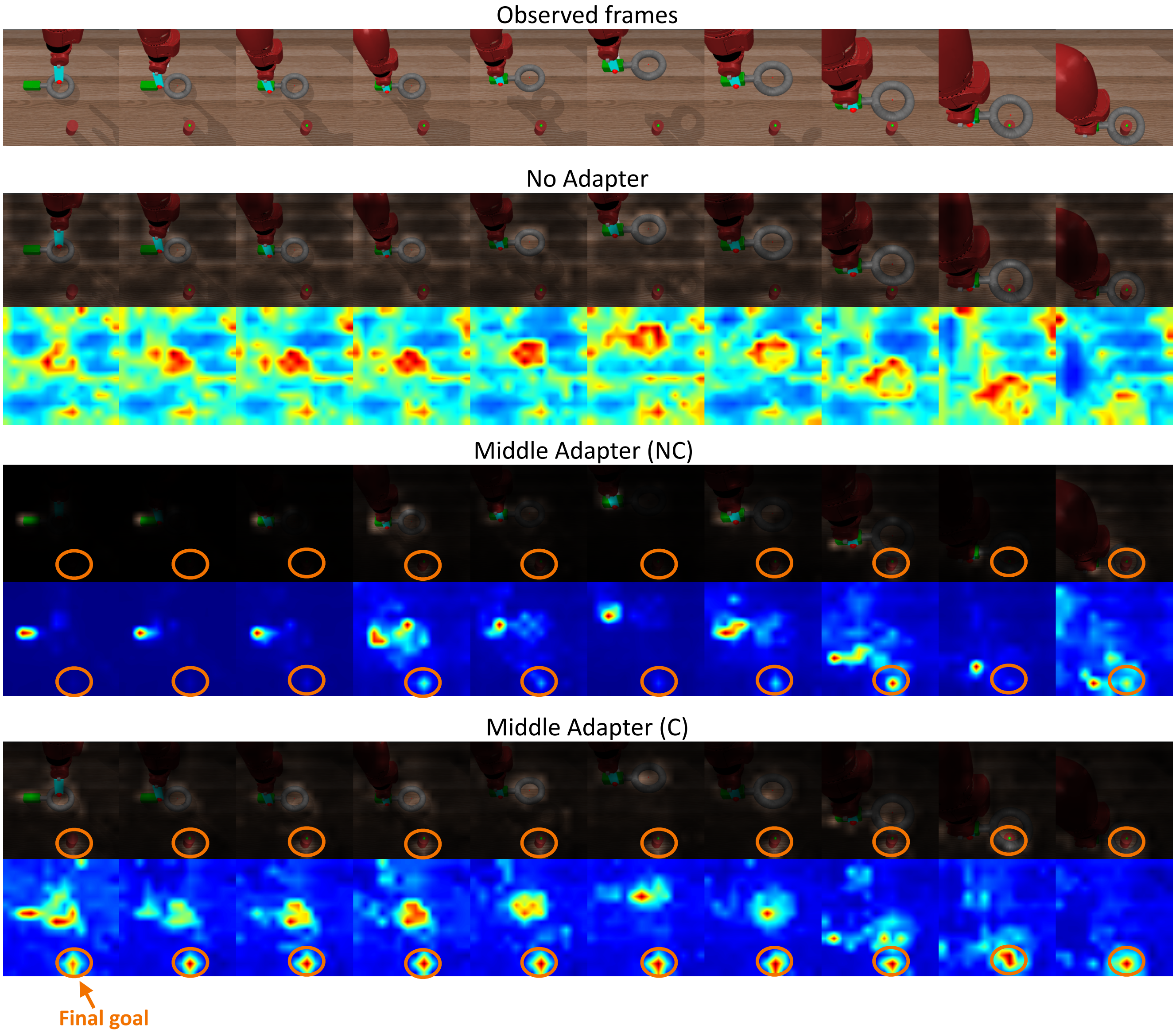

Figure 2. **Visualization of attention maps (Assembly task)**. First row: observed input frames. Following blocks: for each model type, we show the attention map of the last ViT layer, first overlaid on top of the visual frame and below as a colored heatmap. In this example, middle adapters allow to focus the attention on important regions, and task conditioning leads to a better covering of entire objects and agent parts, along with greater attention towards the final goal for all frames.

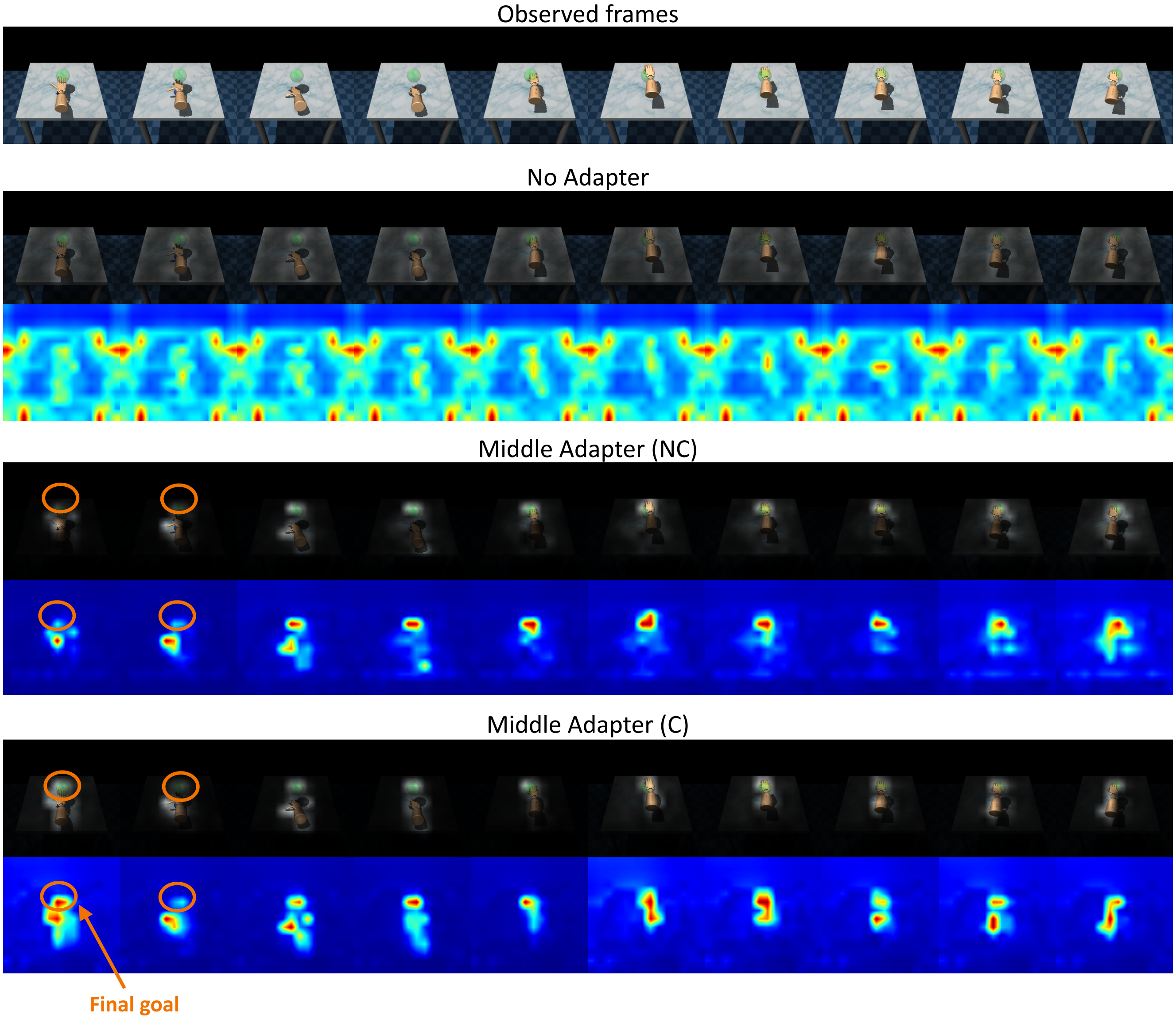

Figure 3. **Visualization of attention maps (Relocate task)**. First row: observed input frames. Following blocks: for each model type, we show the attention map of the last ViT layer, first overlaid on top of the visual frame and below as a colored heatmap. In this example, middle adapters allow to focus the attention on important regions, and task conditioning leads to a better covering of the robotic hand and the sphere goal in all frames.

to follow the setup used to pre-train the model.

Task embedding — the task embedding is a 1024-dim vector. For *known tasks*, it is predicted by a linear embedding layer from a 1-in-K vector where K=12.

Middle adapters — one adaptation module α_l is inserted after each self-attention layer inside ϕ. It is composed of 2 fully-connected layers with respectively 384 and 768 neurons. A *GELU* activation function is applied to the output of the first layer. The input to a middle adapter is the concatenation of the task embedding and a token representation from the previous self-attention layer. It thus processes all tokens as a batch.

Aggregation fully-connected layer — the input to the aggregation fully-connected layer ψ is a concatenation of the 768-dim representation of all 14×14=196 tokens. It is implemented as a simple fully-connected layer predicting a 768-dim vector representation.

Top adapter — the top adapter τ is fed with the output of ψ, again concatenated to the task embedding. It is composed of 2 fully-connected layers that both have 768 neu-

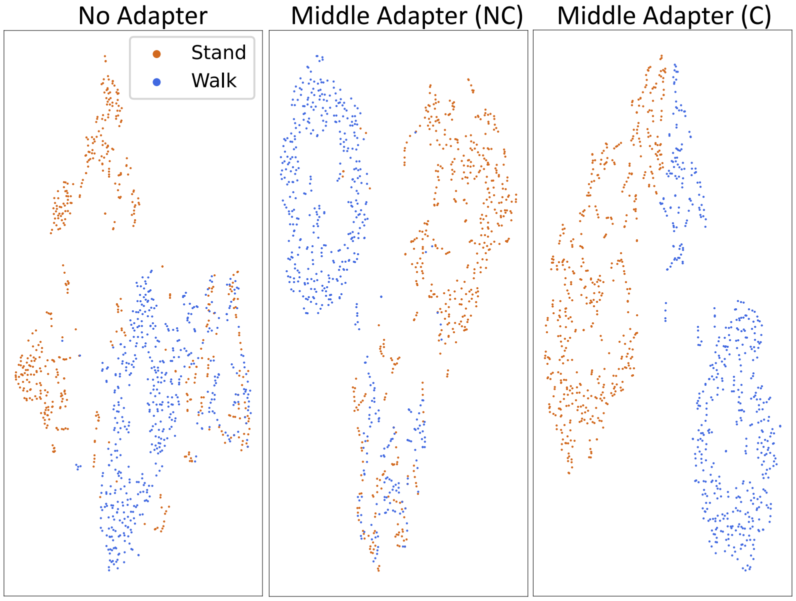

Figure 4. **Task-related information inside visual embeddings**: t-SNE plots of visual embeddings for a set of frames for DMC *Stand* and *Walk* tasks. We chose these tasks for their visual similarity, making it very hard to distinguish between them from vision only. Conditioning of the middle adapters leads to two properly separated clusters, showing the insertion of task-related information into the visual embeddings.

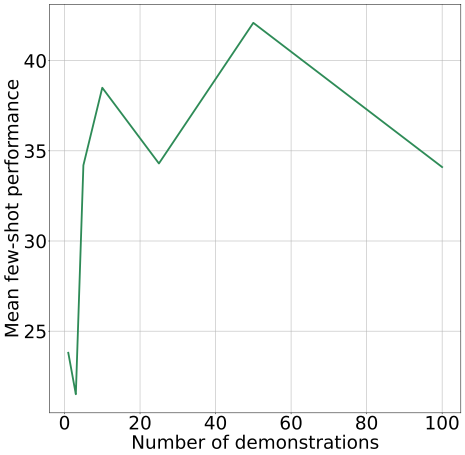

Figure 5. Few-shot — **Impact of the number of demonstrations on few-shot adaptation to unseen tasks**: Few-shot performance as a function of the number of demonstrations used to optimize the task embedding. We can achieve 23.8% from a single demonstration, and more demonstrations can lead to higher performance. However, the trend is not as simple as 100 demonstrations lead to the same performance as 5 demonstrations.

rons. A $ReLU$ activation function is applied to the output of the first layer.

Multi-task policy — The policy π^m is a 3-layer MLP, with 256 neurons for all layers and $ReLU$ activation functions. A batch normalization operation is applied to the input to the policy. π^m outputs a 30-dim action vector, as 30 is the number of components in the action space with the most components among the 12 known tasks. When solving a task with a smaller action space, we mask out the additional dimensions.

I. Impact of the number of demonstrations on few-shot adaptation to unseen tasks

Figure 5 presents the evolution of the average few-shot performance of our method across the 15 unknown tasks depending on the number of available demonstrations when optimizing the task embedding. From only a single demonstration per task, we can already reach a satisfying 23.8% mean performance. Adding more demonstrations can allow to reach higher performance, but the scaling law does not appear to be as simple as using 100 demonstrations leads to the same final performance as 5 demonstrations.