From Single-Task to Multi-Task Reinforcement Learning in Meta-World

Florian Langer 12002209
Marko Petrovic 01527067
Anto Skoro 01607413
Christopher Walderich 12406897

I. Abstract

The aim of this project - part of Lecture *384.195, "Robot Learning"* - was to identify adequate policies for various experiment setups using Meta-World, encompassing three single tasks and two multi-task setups. The Soft Actor-Critic (SAC) deep reinforcement learning algorithm was utilized and extended by strategies for multi-task setups, including one-hot vector encoding, transfer learning, curriculum learning and multi-head critic. Policies with a 100% success rate were found for the three selected individual single tasks. Also, a policy with a success rate of 93.3% for the multitask setup MT3 was found using curriculum learning in combination with one-hot vector encoding. The use of one-hot vector encoding resulted in a 66% success rate in the MT10 setup, which is regarded as the most challenging setup in this examination. The study shows that SAC without any modification can be sufficient for training single tasks. In contrast when training for multi-task setup, sophisticated expansion strategies are proven useful to increase the performance significantly compared to utilizing SAC without any extension.

Keywords: Meta-World, One-hot Encoding, Multi-head Critic, Transfer Learning, Curriculum Learning

II. Introduction

AS mentioned before the ability to learn different tasks simultaneously using reinforcement learning methods has proven to be difficult. When tasks are learned sequentially, catastrophic forgetting is likely to occur. This phenomenon can primarily be attributed to interference in shared representations and non-stationary learning dynamics. As the policy and value function parameters are updated to optimize performance on new tasks, previously learned task-specific features may be overwritten.
This work uses **Meta-World** [2] to investigate training strategies for single- and multi-task setups. Meta-World provides standardized task suites ranging from single-task learning - further referenced by MT1 benchmarks - to larger and more diverse multi-task manipulation benchmarks such as MT10. Based on this, a custom MT3 benchmark is also defined. This benchmark combines the *reach-v3*, *push-v3*, and *pick-and-place-v3* tasks into one scenario, which should also be investigated individually.
The Soft Actor-Critic (SAC) algorithm offers several key advantages compared to other reinforcement learning algorithms, including high sample efficiency due to off-policy learning and extensibility to curriculum and transfer learning. Its effectiveness is demonstrated by the promising results reported in the Meta-World paper [2]. Hence, SAC is deployed here for further investigations. The baseline framework **Stable Baselines3** [3] is used for vectorization, which is required for multi-task setups.
All single-task and multi-task setups shall be investigated using a random hyperparameter search. Additionally, advanced training strategy shall be deployed to overcome the previously mentioned issues of learning multi-tasks simultaneously. A task-conditioned policy can address the complexity finding a generalized policy. Multiple critics can be utilized to estimate the Q-value separately for each environment, as an alternative for overcoming representation and generalization issues. To tackle the challenge of individual, highly specialized, and complex tasks being omitted during the learning process, the curriculum learning approach, potentially in conjunction with transfer learning, aims to learn additional tasks in-depth, step by step, without losing sight of previously acquired knowledge and reusing it. This can also speed up training.
These methods are described conceptually in more detail in Section III. More insights into the actual training and results are given in Section IV. Finally, conclusions are drawn in Section V.

III. Methodology

This section shall mainly provide an overview of all applied training strategies during the project. A quick, structured overview, in Figure 1, helps you understand all the methods and combinations applied to the different task setups.
The simplest strategy is to apply the default SAC algorithm to the MT1-, MT3-, and MT10-setups. A hyperparameter search was performed using random search. While finding the best set $\mathcal{H}_{MT1}$ for the MT1 setups, all hyperparameters $\mathcal{H}$ were taken into account for investigation. Excellent results with 100% accuracy in all single tasks were already obtained in the previous assignments and reused here. Based on these results and experiences while finding $\mathcal{H}_{MT1}$, a subset of $\mathcal{H}$ was identified that appeared to have the greatest impact on training: training steps, activation function, training frequency, learning

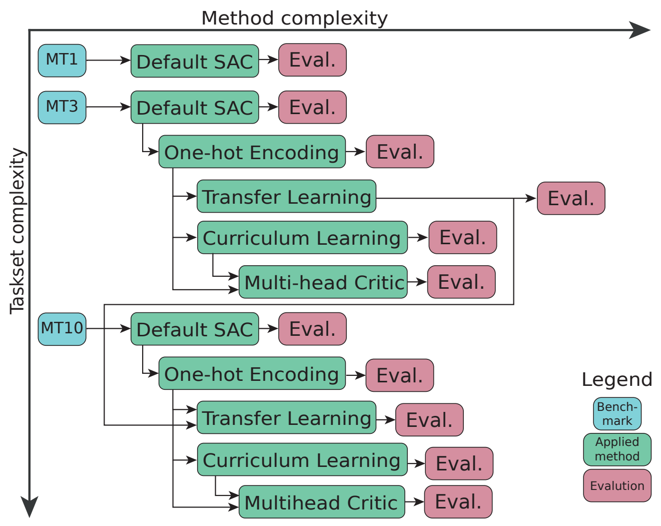

Fig. 1: Applied methodology workflow

rate (*lr*), buffer size (*buffer*), batch size (*batch*), gradient steps (*gr-steps*), and MLP network architecture (*net-arch*). All other parameters were taken from the Meta-World paper [2].
Additionally, advanced methods are investigated for multitask training setups. In the following the different methods applied to the two different setups are further explained.

A. MT3-Setup

In multitask reinforcement learning the aim is to learn a single policy $\pi(a|s)$ which performs optimal on M-different tasks $\{\mathcal{T}\}_{i=1}^{M}$ regarding the relevant reward function γ_i. Conditioning the policy on each task $\pi(a|s,z)$, where z is an encoding of the task ID (**One-hot Encoding**), provides the model with more information. This can help with learning task-specific execution policies for the currently assigned task while still using shared manipulation skills with other tasks. Thus, it can help with generalizing a single learned policy to multiple tasks. In this setup the task-conditioning is implemented using one-hot encoding of all tasks, which is appended to the current observation space. However, it has to be mentioned a disadvantage of one-hot-encoding is its fixed set of tasks. Each task corresponds to a separate dimension. Therefore, changing the number of tasks would also change the dimensionality of the observation space.
Furthermore, using only one critic estimating the Q-value function of M different tasks $Q(a,s,z)$, can lead to poor ratings of state-action pairs, since actions are valued differently among all tasks. This can prevent the model from learning the optimal policy, as actions are evaluated incorrectly. Rather than task-conditioning the observation state, a separate critic can be implemented for each task during training to evaluate state-action pairs separately (**Multi-head Critic**). Moreover, single-task training showed that the reach-v3 task is learned more easily than the push-v3 and pick-and-place-v3 tasks.
Curriculum Learning enables the successful learning of more difficult task sets that would otherwise be learned poorly in a standard multi-task training setting by following a predefined curriculum, consisting of a specific constructed and ordered set of stages containing various tasks. The curriculum used here will be referred to as **Curriculum V1** and begins with learning in a vectorized environment containing equally distributed tasks. At the end of each training session, the tasks are identified which are omitted during training and show no learning progress. With that the distribution is adjusted, so that the identified tasks are overrepresented to intensify the training of them. This can be repeated. Other investigated curriculums are mentioned in subsection Subsection III-B.
Finally, another promising approach is to use the results obtained from the single task and fine-tune them for the MT3-setup via **Transfer Learning**. This can rapidly speed up the training process by utilzing previously gathered knowledge. In the chosen Transfer Learning method, the whole network is further trained with a lowered learning rate. The task *pick-and-place-v3* is by far the most difficult one among the MT3-tasks. This led to the decision to fine-tune and adapt the *pick-and-place-v3* specific model to the remaining tasks. One could argue that this strategy is a combination of Transfer Learning and Curriculum Learning. In this work, however, it is referred to as Transfer Learning since the MT1-setup is adapted directly to the MT3-setup without any intermediate steps.
While working on the project, the lack of task-conditioning was identified as one of the biggest bottlenecks of multitask reinforcement learning as it can be obtained from Figure 4. Adding task-conditioning already improved success rates significantly, documented in Section IV. This is why the observation state is always task-conditioned using one-hot task encoding when investigating further advanced extensions in the multitask training setup. For example, fusing multi-head critic and task-conditioned policy for even better task-conditioning and separated representation. This holds also for the MT10-setup which is explained in the following.

B. MT10-Setup

The main concepts and strategies of advanced methods are the same as before. In particular, the concepts of task conditioning and Multi-head Critic are unchanged and only require scaling to additional tasks. However, strategies such as curriculum learning and transfer learning were expanded or applied slightly differently. In addition to the **Curriculum V2**, introduced in Subsection III-A, a progressive curriculum is implemented. In this case multiple phases are predefined, which contain different task environments. They can be customized arbitrarily using a configuration file. A threshold must be defined for each phase. Once this threshold is reached within a phase, training progresses to the next one defined in the curriculum. This means that training on a subset of tasks must be sufficiently successful before new tasks are added. Two predefined variations of progressive curriculum are already implemented naively in the code. One adds tasks one by one gradually ordered by difficulty (**Sequential Curriculum**). The other one defines seven phases, which can contain more than one task and increases the difficulty the higher the phase is (**Mixed Curriculum**). The following work refers to all kinds of progressive curricula as Curriculum V2. Further, rather than adapting a single task model directly to the MT10-setup using Transfer Learning, a found solution for MT3 shall be fine-tuned for MT10. This is also combined with curriculum V2 to introduce some intermediate steps towards learning ten tasks.

IV. Results

This section summarizes relevant results and compares the most successful approaches with their best-found configurations.

A. MT1 Single-Task Learning

Using the code provided in the references[1], MT1 policies with a **100% success rate** were successfully obtained for the selected single-task setups (*reach-v3*, *push-v3*, *pick-and-place-v3*) by appropriately adjusting the training hyperparameters, which are documented in Table A.1. The learning difficulty varied between the considered tasks, which required task-dependent tuning of the total number of training time steps and the number of parallel environments. In Figure 2 the achieved reward scores are plotted against the total number of time steps.

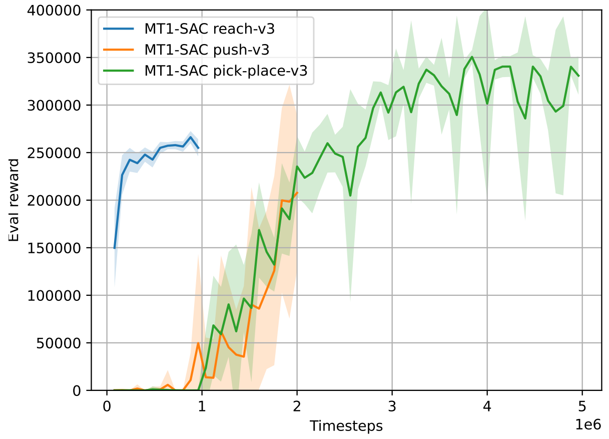

Fig. 2: MT1 Evaluation - Reward scores and corresponding standard deviations

In particular, more complex manipulation tasks like *pick-and-place-v3* required longer training durations and a higher number of parallel environments in order to provide the agent with sufficient exploration opportunities compared to *reach-v3* until convergence of the rewards score occurs.

For this training stage, no advanced multitask extensions such as One-hot Encoding, Curriculum Learning, or Multi-head Critics were required. The standard Soft Actor-Critic algorithm proved sufficient for learning individual tasks, provided that the hyperparameters were carefully adapted to the respective task difficulty. These results confirm that SAC without architectural or algorithmic extensions is sufficient for solving single-task Meta-World environments when appropriate hyperparameter tuning is applied.

B. MT3 Multi-Task Learning

Achieving successful MT3 policies required more sophisticated strategies. As shown in Figure 3, applying only the default SAC (hereafter referred to as the default case) resulted in poor success scores. First, One-hot Encoding was added to the algorithm stack, which doubled the average success rate compared to the default case. This improvement is also reflected in the reward evaluation depicted in Figure 4.

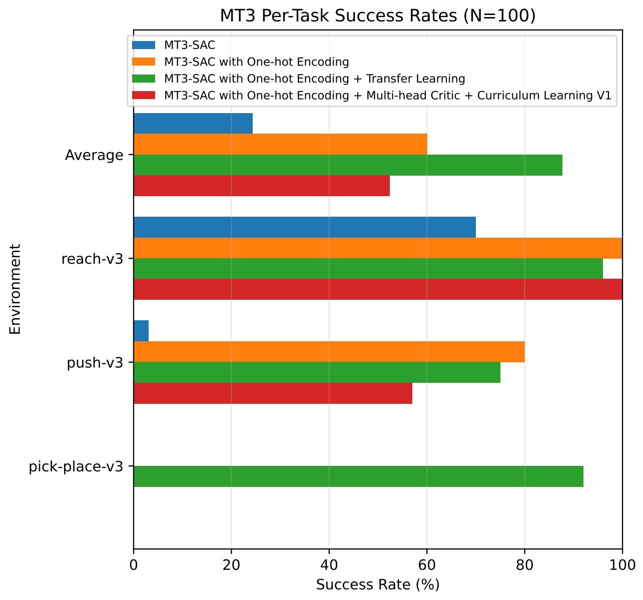

Fig. 3: MT3 - Per-task and average success rates

An alternative approach we reviewed is one-hot encoding in combination with transfer learning. As a starting point, the trained model for *pick-and-place-v3* was chosen, as this learned policy was the most specific and therefore a suitable basis for generalization. After training for 5 million time steps on this single task, the environment was replaced by a vectorized environment including all MT3 tasks. This transition is marked by a significant drop in the reward score. After training for only 1 million additional time steps with a learning rate lowered by a factor of 0.1, the average success rate approximately tripled compared to the base case (see Figure 4).

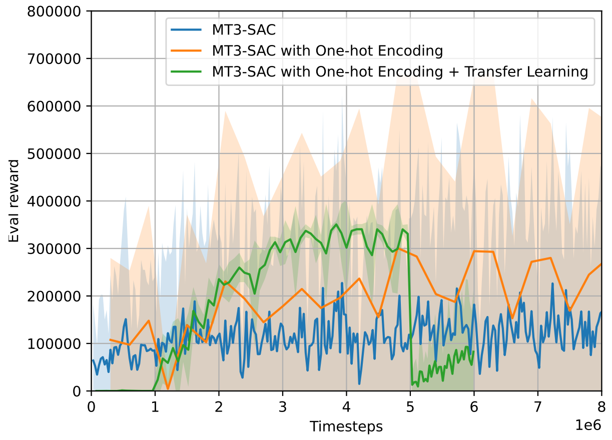

Fig. 4: MT3 Evaluation - Reward scores and corresponding standard deviations

Finally, a fourth approach achieved reasonable results, however, it performed not as good as the previously described one. We combined a One-hot Encoding, with Multi-head Critic and Curriculum Learning V1 (for details see Section III). Here, the major problem was again the complexitiy of learning the *pick-and-place-v3* task.

Lastly, an important remark regarding the notably large standard deviation shown in Figure 4: the *reach-v3* task typically yields high reward values due to its relative ease, whereas the *pick-and-place-v3* produces very low or even negative rewards because of its difficulty. This results in a substantial discrepancy between the individual task rewards. The two extreme values inflate the standard deviation due to their large separation. This explanation can also be applied to the following Figure 6.

C. MT10 Multi-Task Learning

For the MT10 multi-task setup, several training strategies were evaluated using the Soft Actor-Critic algorithm, including standard SAC without task-specific extensions, SAC with One-hot Encoding, and SAC with a combination of One-hot Encoding and a Multi-head Critic architecture. Among all tested approaches, the best performance was achieved using one-hot task encoding without a multi-head critic as it can be retrieved from Figure 5. Using this method, a maximum overall success rate of **66%** was obtained after 27 million training time steps.

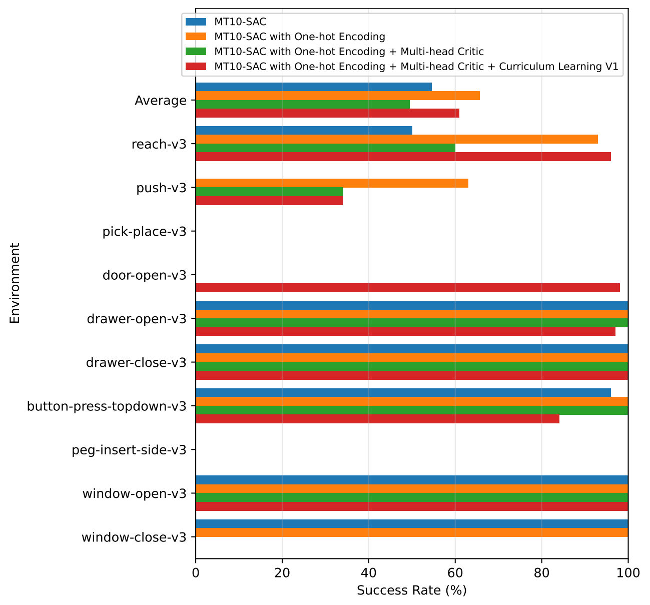

Fig. 5: MT10 - Per-task and average success rates

To investigate whether further improvements were possible, training was continued for an additional 9 million time steps, resulting in a total of 36 million time steps. However, no further increase in the success rate was observed, and the performance remained at **66%**, indicating convergence of the learned policy.

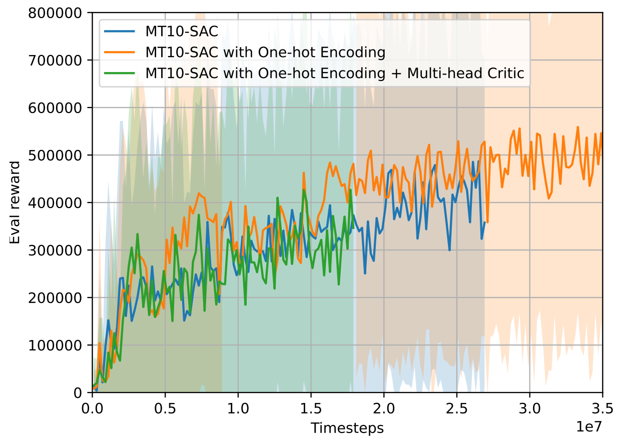

Fig. 6: MT10 Evaluation - Reward scores and corresponding standard deviations

The final evaluation of this training run, conducted over 100 episodes, yielded an average reward of approximately 5.15×10^5 with a success rate of **66/100 episodes**. A per-task analysis revealed strong performance on several tasks, such as *drawer-open-v3*, *drawer-close-v3*, *button-press-topdown-v3*, *window-open-v3*, and *window-close-v3*, each achieving success rates close to or equal to 100%. In contrast, more complex manipulation tasks, most notably *pick-place-v3* and *door-open-v3*, consistently failed, achieving a success rate of 0%. This highlights the significant task imbalance present in the MT10 benchmark and illustrates the difficulty of learning a single policy that performs well across all tasks.

In comparison, the combination of One-hot Encoding with a Multi-head Critic resulted in inferior performance. After 9 million training timesteps, this approach achieved an overall success rate of **48%**, which further decreased to **46%** after 18 million training steps, indicating no further improvement with more training. Moreover, the only method that achieved good results in *door-open-v3* was curriculum learning, but no successes were seen in *window-close-v3*. Training runs with default SAC yielded lower success rates than any used advanced multitask extension.

V. Discussion

Due to partial loss of TensorBoard logs, our results and discussion are limited to the data available in this work. The conclusion that are drawn might not be extensible to all experiments and are therefore specific to our provided figure.

A. Interpretation of results

For single-task setups, standard algorithms may be sufficient for successful training. Even a very demanding task, such as *pick-and-place-v3* can be completed with a 100% success rate, provided that an extensive hyperparameter search of the SAC algorithm without any further measure is performed. Thus, reinforcement learning models have great potential to learn specific tasks with sufficient training and efficient data sampling. Using shorter episodic lengths appeared to be helpful when learning more difficult tasks. Longer episodes carry the risk that the agent will diverge further from the goal. Especially in the case of difficult tasks, the agent will rarely achieve partial successes with positive rewards. This results in a poor training dataset with mostly useless data far off the goal. With shorter episodes, more initial behaviors are sampled, allowing for quicker convergence to optimal behavior with regard to the reward function for solving the task. Additionally, intensive training on all available data using gradient steps equal to -1 appears necessary for a significant increase in performance.

However, multitask setups require models that excel at solving various tasks. This makes training much more difficult and requires far more steps. Sample efficiency becomes much more important. One-hot Encoding compared to default SAC showed fast convergence and yielded high average reward levels in both the MT3 and MT10 setups after training, see Figure 4 and 6. This shows the efficiency of applying a task-conditioned policy. High standard deviation in all curves show that some tasks are not learned, yielding high discrepancy in reward. Analysing Figure 6, the reward learning curve of Multi-head Critic seems to not achieve a higher reward level than default SAC and also shows no convergence yet.

Transfer learning appears to be an efficient strategy, particularly for the MT3 setup. Compared to the other strategies, the trained model achieves significantly better performance with a fraction of the training steps. It indicates a great sample efficiency and safes runtime. While the other strategies completely neglect “pick-and-place,” transfer learning achieves a 92% success rate here and still achieves very good results in “push” and “reach”. When adapting to the MT3 setup, the average reward, in Figure 4, experiences a sharp drop and then rises slowly to only a fraction of the previous level. Further, the deviation increases with increasing training steps when applying Transfer Learning. This suggests that the model did not learn much after transfer learning was applied, and may even starts to forget tasks indicated by the increasing deviation.

A comparison with the original MT1 model for ”pick-and-place” reveals that it achieves an even better result: a 98.2% success rate over all MT3 tasks. This is better than the success rate of the transfer learning model that was ultimately further trained. It concludes that the complex ”pick-and-place” task provides already all skills needed for ”reach” and ”push” to be solved. This can be an explanation why Transfer Learning worked only poorly in the MT10 setup. The task set was too diverse, so that knowledge from ”pick-and-place” was no longer sufficient and more extensive training would have been required to learn new skills.

Regarding the MT10 benchmark, two key observations can be highlighted. First, by employing a curriculum-based approach, the actor was able to learn tasks (here *door-open-v3*) that were not successfully learned using alternative architectural approaches. Second, the curriculum phases must be designed with care in advance. If task weighting or scheduling is not properly balanced, tasks that are comparatively easier to learn (here *window-close-v3*) may either dominate the learning process or, conversely, previously acquired tasks may be forgotten. Thus, curriculum learning can be considered a powerful mechanism for training in multi-task environments, provided that the curriculum stages are constructed deliberately and systematically. However, no reliable statement regarding convergence speed or stability can be made due to the previously mentioned loss of training logs.

B. Conclusion

All multi-task training results demonstrate that One-hot Encoding is a simple, yet highly effective and robust extension for multitask reinforcement learning, significantly outperforming both unmodified SAC and more complex architectures such as the Multi-head Critic in this context. Curriculum learning seems to be a sufficient way to focus learning on different tasks. Nevertheless.

the tasks *pick-place-v3* and *peg-insert-side-v3* have never been learned in the setup.
However, this generalization statement has to be handled with caution, as there might be scenarios where Multi-head Critics outperform task conditioning. Multi-head Critic decreases task interferences in the critic training. Actions of the actor and their potential future reward are estimated better, containing the potential of converging to a better policy. This does not necessarily mean that training converges faster. At the same time, the observed performance saturation suggests that additional mechanisms, such as more refined Curriculum strategies or task-specific loss balancing, may be required to further improve performance on the most challenging tasks. For that, the average reward can be used to weight the balance task-specific loss, where low average reward indicates tasks being omitted. This also can help to avoid the high discrepancy between task rewards, mentioned in Subsection IV-C

References

[1] F. Langer, M. Petrovic, A. Skoro, and C. Walderich. "Meta-world multi-task training framework," Accessed: Jan. 23, 2026. [Online]. Available: https://github.com/sirknival/robot_learning_project.

[2] T. Yu et al., "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," *CoRR*, 2019. arXiv: 1910.10897. [Online]. Available: http://arxiv.org/abs/1910.10897.

[3] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html.

Appendix A
MT1 - Training Details

TABLE A.1: Training Hyperparameters for MT1-benchmarks

Hyperparameter	**reach-v3**	**push-v3**	**pick-and-place-v3**
Random seed		42	
Algorithm		SAC	
Policy		MlpPolicy	
Learning rate		1×10^{-4}	
Learning starts		5,000	
Batch size		256	
Soft update coefficient (τ)		0.005	
Discount factor (γ)		0.99	
Train frequency		1	
Gradient steps		-1 (all available data)	
Entropy coefficient		auto	
Target entropy		auto	
State-dependent exploration		False	
Policy network architecture		[256, 256, 256]	
Activation function		ReLU	
Initial log standard deviation		-3	
Reward normalization		False	
Replay buffer size	1e6	1.5e6	1.5e6
Total timesteps	1×10^6	2×10^6	5×10^6
Max episode length	500	200	200
Number of parallel environments	8	8	10

Appendix B
MT3 - Training Details

TABLE B.1: Shared Training Hyperparameters in the found MT3-Models

Hyperparameter	
Random seed	42
Algorithm	SAC
Batch size	256
Soft update coefficient (τ)	0.005
Discount factor (γ)	0.99
Train frequency	1
Entropy coefficient	auto
Target entropy	auto
State-dependent exploration	False
Activation function	ReLU
Initial log standard deviation	-3
Reward normalization	False

TABLE B.2: Varying Training Hyperparameters used in the found MT3-Models

Hyperparameter	**(1)**	**(2)**	**(3)**	**(4)**
Policy	MlpPolicy	MlpPolicy	MlpPolicy	Multi-head Critic
Learning rate	3×10^{-4}	3×10^{-4}	1×10^{-5}	1×10^{-4}
Replay buffer size	1×10^{6}	1×10^{6}	1.5×10^{6}	1×10^{6}
Learning starts	5,000	5,000	5,000	10,000
Gradient steps	1	1	-1	1
Policy network architecture	[512, 1024 1024, 512]	[512, 1024 1024, 512]	[256, 256, 256]	[256, 256, 256]
Total time steps	12×10^{6}	12×10^{6}	1×10^{6} (6×10^{6})	33×10^{6}
Max episode length	500	500	200	500

(1) - SAC
(2) - SAC with One-hot Encoding
(3) - SAC with One-hot Encoding + Transfer Learning
(4) - SAC with One-hot Encoding + Multi-head Critic + Curriculum V1

TABLE B.3: Used learning schedule in the MT3-setup for *SAC with One-hot Encoding + Multi-head Critic + Curriculum V1* expressed as task distribution in [%] at each phase

Phase	**P0**	**P1**	**P2**	**P3**	**P4**	**P5**
Time steps	9×10^{6}	3×10^{6}	3×10^{6}	3×10^{6}	3×10^{6}	12×10^{6}
reach-v3 [%]	50.0	20.0	0.0	13.3	13.3	6.7
push-v3 [%]	50.0	70.0	100.0	53.3	40.0	33.3
pick-place-v3 [%]	0.0	10.0	0.0	33.3	46.7	60.0

Appendix C
MT10 - Training Details

TABLE C.1: Shared Training Hyperparameters in the found MT10-Models

Hyperparameter	
Random seed	42
Algorithm	SAC
Replay buffer size	1×10^6
Batch size	256
Soft update coefficient (τ)	0.005
Discount factor (γ)	0.99
Train frequency	1
Entropy coefficient	auto
Target entropy	auto
State-dependent exploration	False
Activation function	ReLU
Initial log standard deviation	-3
Max episode length	500
Reward normalization	False

TABLE C.2: Varying Training Hyperparameters used in the found MT10-Models

Hyperparameter	**(1)**	**(2)**	**(3)**	**(4)**
Policy	MlpPolicy	MlpPolicy	Multi-head Critic	Multi-head Critic
Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-5}	1×10^{-4}
Learning starts	5,000	5,000	5,000	10,000
Gradient steps	1	1	1	5
Policy network architecture	[512, 1024 1024, 512]	[512, 1024 1024, 512]	[512, 1024 1024, 512]	[256, 256, 256]
Total time steps	27×10^6	27×10^6	18×10^6 (6×10^6)	24×10^6

(1) - SAC
(2) - SAC with One-hot Encoding
(3) - SAC with One-hot Encoding + Multi-head Critic
(4) - SAC with One-hot Encoding + Multi-head Critic + Curriculum V1

TABLE C.3: Used learning schedule in the MT10-benchmark for *SAC with Task Conditioning + Multi-head Critic + Curriculum V1* expressed as task distribution in [%] at each phase

Phase	**P0**	**P1**	**P2**	**P3**	**P4**	**P5**
Time steps	3×10^6	3×10^6	6×10^6	3×10^6	6×10^6	3×10^6
reach-v3 [%]	10.0	6.7	3.3	3.3	3.3	10.0
push-v3 [%]	10.0	13.3	16.7	33.3	30.0	6.7
pick-place-v3 [%]	10.0	13.3	16.7	6.7	3.3	0.0
door-open-v3 [%]	10.0	13.3	16.7	10.0	6.7	10.0
drawer-open-v3 [%]	10.0	6.7	3.3	3.3	3.3	10.0
drawer-close-v3 [%]	10.0	6.7	3.3	3.3	3.3	10.0
button-press-topdown-v3 [%]	10.0	6.7	3.3	3.3	3.3	10.0
peg-insert-side-v3 [%]	10.0	13.3	16.7	6.7	10.0	0.0
window-open-v3 [%]	10.0	6.7	3.3	3.3	3.3	10.0
window-close-v3 [%]	10.0	13.3	16.7	26.7	33.3	33.3