Supplementary Material

A. Megaverse: technical details

A.1. Interface

Megaverse platform provides a vectorized version of the OpenAI Gym interface (Brockman et al., 2016) for interacting with the environments. This is a natural extension of the standard Gym interface for parallel simulators: since a single Megaverse instance simulates experience for M agents in N environments per step, the STEP() function accepts a vector of $N \times M$ actions, and returns a vector of $N \times M$ observations, rewards, and episode termination flags. The only difference with the original OpenAI Gym design is that Megaverse environments do not require RESET() calls, except right after initialization. Individual simulated environments can have different episode durations, so we reset these individual environments automatically instead of relying on external code to perform resets. This has an additional performance benefit: we don't have to synthesize the last observation in the episode which is never seen by the agents.

Although Megaverse core engine is written in C++, the high-level interface is also available through Python bindings (Jakob et al., 2017).

A.2. Observation and action spaces

Megaverse provides mechanisms to configure custom observation and action spaces for individual scenarios, although all Megaverse-8 scenarios use the same unified observation and action spaces to streamline and simplify the experimentation. The observations are provided as 128×72 RGB images, and this is the only sensory input received by the agents. On top of the synthesized views of the 3D world, the observations can also contain additional information about the environment. We implement simple pixel-space GUI consisting of geometric primitives rendered in the agent's field of view. These can play the role of various bars and indicators, such as health bars, or team affiliation flags for team-based multi-agent scenarios. In Megaverse-8 scenarios we only use this GUI to notify agents about the remaining time in the episode.

Table A.1 describes agent's affordances. At each step the agents can independently choose walking and gaze directions, and whether they choose to jump or interact with an object. OpenAI Gym represents this action space using a tuple of discrete action spaces: TUPLE(DISCRETE(3), DISCRETE(3), DISCRETE(3), DISCRETE(3), DISCRETE(2), DISCRETE(2)). In our implementation the policy networks outputs six distinct probability vectors, which we interpret as independent categorical action distributions, although the action space can also be flattened into a single discrete action space with 324 options.

Action head	**Number of actions**	**Comment**
Moving	3	no-action / forward / backward
Strafing	3	no-action / left / right
Turning	3	no-action / turn left / turn right
Vertical gaze direction	3	no-action / look up / look down
Jumping	2	no-action / jump
Object interaction	2	no-action / interact
Total number of possible actions	324	

Table A.1. Megaverse-8 action space.

B. Megaverse-8

Please refer to the project website for detailed videos demonstrating the environments: `www.megaverse.info`.

B.1. Reward functions

Table B.1 describes the reward functions in Megaverse-8 scenarios, as seen by the learning algorithm. Besides the dense rewards that facilitate learning and exploration, Megaverse-8 environments also provide a single sparse reward (true objective) per episode that measures the real progress on the task. In all environments except TowerBuilding the true objective takes the value $+1$ when the task is successfully completed and 0 otherwise. In the TowerBuilding scenario the true objective is to maximize the height of the structure built during the episode.

In addition to task completion (true objective) results reported in the main paper, we also report dense rewards achieved by the agents in our experiments, see Figures C.1 and C.2.

Scenario	Dense reward	True objective
ObstaclesEasy ObstaclesHard	$+1$ reached target location $+0.5$ collected a green diamond $+5$ all agents reached the target	$+1$ (success) all agents reached the target 0 (failure) episode timed out
Collect	$+1$ collecting green diamond -1 collecting red diamond $+5$ collected all green diamonds -0.5 agent fell into the void	$+1$ (success) collected all green diamonds 0 (failure) episode timed out
Sokoban	$+1$ moved box onto target -1 moved box from the target $+10$ moved all boxes to targets	$+1$ (success) moved all boxes to targets 0 (failure) episode timed out
HexExplore	$+5$ found a pink diamond	$+1$ (success) found a pink diamond 0 (failure) episode timed out
HexMemory	$+1$ collected a matching object -1 collected a non-matching object	$+1$ (success) collected all matching objects 0 (failure) episode timed out
Rearrangement	$+1$ moved object to a correct position $+10$ all objects in correct positions	$+1$ (success) all objects in correct positions 0 (failure) episode timed out
TowerBuilding	$+0.1$ entered building zone with an object $+0.05(h + 2^h)$ placed an object in the building zone h - height at which the object was placed	$+h_{max}$ where h_{max} is the max height of the tower

Table B.1. Megaverse-8 scenarios dense rewards and final objectives.

C. Experimental details

C.1. Performance analysis

Table C.1 provides information about hardware configuration of systems used for performance measurements. We focused on commodity hardware commonly used for deep learning experimentation.

While in the main paper we report performance figures measured only in ObstaclesHard scenario, table C.2 provides information about sampling throughput in all Megaverse-8 environments. Values represent the sampling throughput averaged over three minutes. In order to conduct the measurements we used a number of parallel Megaverse processes equal to the number of physical CPU cores with 64 environments simulated in parallel in each process. Performance varies because different scenarios generate environments with different number of geometric primitives and interactive objects. HexMemory and HexExplore environments are based on hexagonal mazes and therefore cannot benefit from the voxel grid based optimizations that allow fast collision checking based on axis-aligned bounding boxes.

C.2. RL experiments: setup and parameters

In all experiments in the paper we used asynchronous PPO (APPO) implementation provided by Sample Factory (Petrenko et al., 2020). Unless stated otherwise, all experiments use Action Conditional Contrastive Predictive Coding (CPC|A) (Guo et al., 2018). For the policy network we use a small convnet model similar to VizDoom model in (Petrenko et al., 2020) with a 2-layer GRU core (Cho et al., 2014). Table C.3 lists the learning algorithm hyperparameters.

	System #1	System #2	System #3
Processor	AMD Ryzen 9 3900X	Intel Xeon Gold 6154	Intel Xeon Platinum 8280
Base frequency	3.8 GHz	3.0 GHz	2.7 GHz
Physical cores	12	36	48
Logical cores	24	72	96
RAM	64 GB	256 GB	320 GB
GPUs	1 x NVidia RTX3090	4 x NVidia RTX 2080Ti	8 x NVidia RTX 2080Ti
GPU memory	24GB GDDR6x	11GB GDDR6	11GB GDDR6
OS	Arch (Jan 2021, Rolling)	Ubuntu 18.04 64-bit	Ubuntu 18.04 64-bit
GPU drivers	NVidia 460.32.03	NVidia 440.95.01	NVidia 450.102.04

Table C.1. Hardware configurations used for performance measurements (training and sampling performance).

Scenario	**Simulation throughput, obs/sec**
ObstaclesEasy	1.27×10^6
ObstaclesHard	1.15×10^6
Collect	8.55×10^5
Sokoban	1.16×10^6
HexExplore	6.5×10^5
HexMemory	5.9×10^5
Rearrange	1.28×10^6
TowerBuilding	1.22×10^6

Table C.2. Sampling throughput in Megaverse-8 scenarios measured on System #3 (8-GPU node).

C.3. Additional RL experiments

To further investigate the training performance of RL agents on Megaverse-8 tasks we conduct a series of additional experiments. First, we extended the training of the APPO+CPC|A agent in single-agent Megaverse-8 environments to 10^{10} environment steps (Figure C.3). Except in TowerBuilding, we did not see a significant increase in agent's performance, which suggests that Megaverse-8 remains a challenging benchmark even in virtually unlimited sample regime (note that training for 10^{10} frames in Megaverse is equivalent to training for 4×10^{10} frames in DeepMind Lab or Atari due to frameskip). Instead of insufficient data, the agents are limited by their exploration abilities and cognitive capacity of relatively simple models. Thus Megaverse-8 environments can be a promising test bed for advanced exploration algorithms and policy architectures.

We also evaluated a single APPO+CPC|A agent trained on all eight Megaverse-8 environments simultaneously (Figure C.4). The agent was trained on a total of 2×10^9 frames of experience, which is equivalent to 2.5×10^8 frames on each of the environments. The results demonstrate that positive transfer can be challenging due to the diversity of Megaverse-8 tasks, although ultimately, combining experience from a diverse set of tasks and incorporating curricula can be instrumental in training capable multipurpose intelligent agents.

Learning rate	10^{-4}
Action repeat (frameskip)	1 (no frameskip)
Framestack	No
Discount γ	0.997
Optimizer	Adam (Kingma & Ba, 2015)
Optimizer settings	$\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-6}$
Gradient norm clipping	1.0
Num parallel Megaverse processes	6
Num envs simulated per process	80
Total number of environments	480
Rollout length T	32
Batch size, samples	2048
Number of training epochs	1
V-trace parameters	$\bar{\rho} = \bar{c} = 1$
PPO clipping range	$[1.1^{-1}, 1.1]$
Exploration loss coefficient	0.001
Critic loss coefficient	0.5

Table C.3. Hyperparameters in Megaverse-8 experiments.

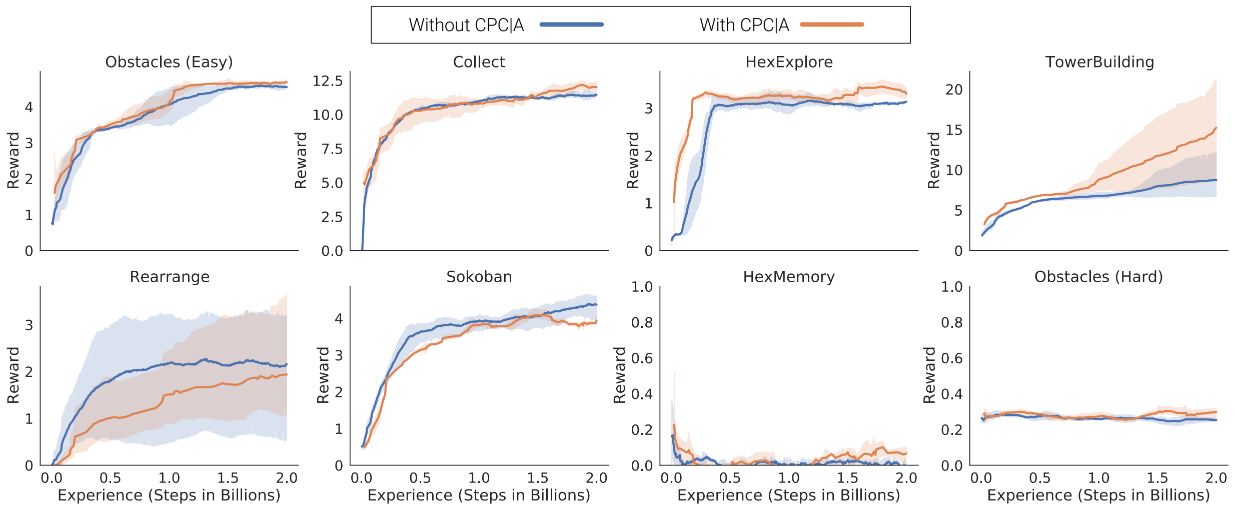

Figure C.1. Total episodic reward achieved by the agents in single-agent scenarios (see reward shaping scheme in Table B.1). Here the results are averaged over three random seeds.

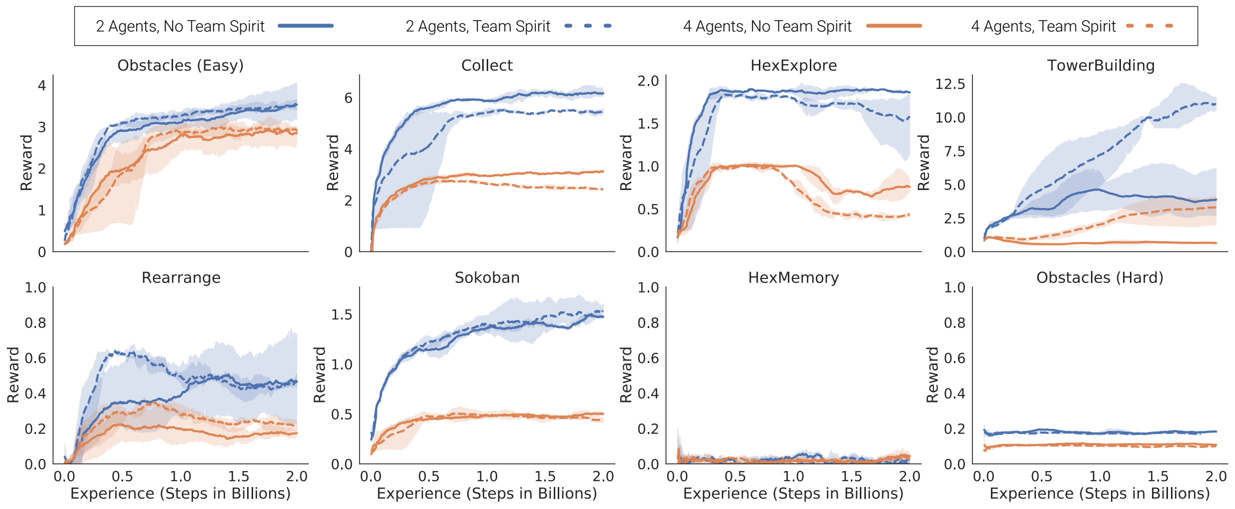

Figure C.2. Total episodic reward achieved by the agents in multi-agent scenarios (see reward shaping scheme in Table B.1). Here the results are averaged over three random seeds.

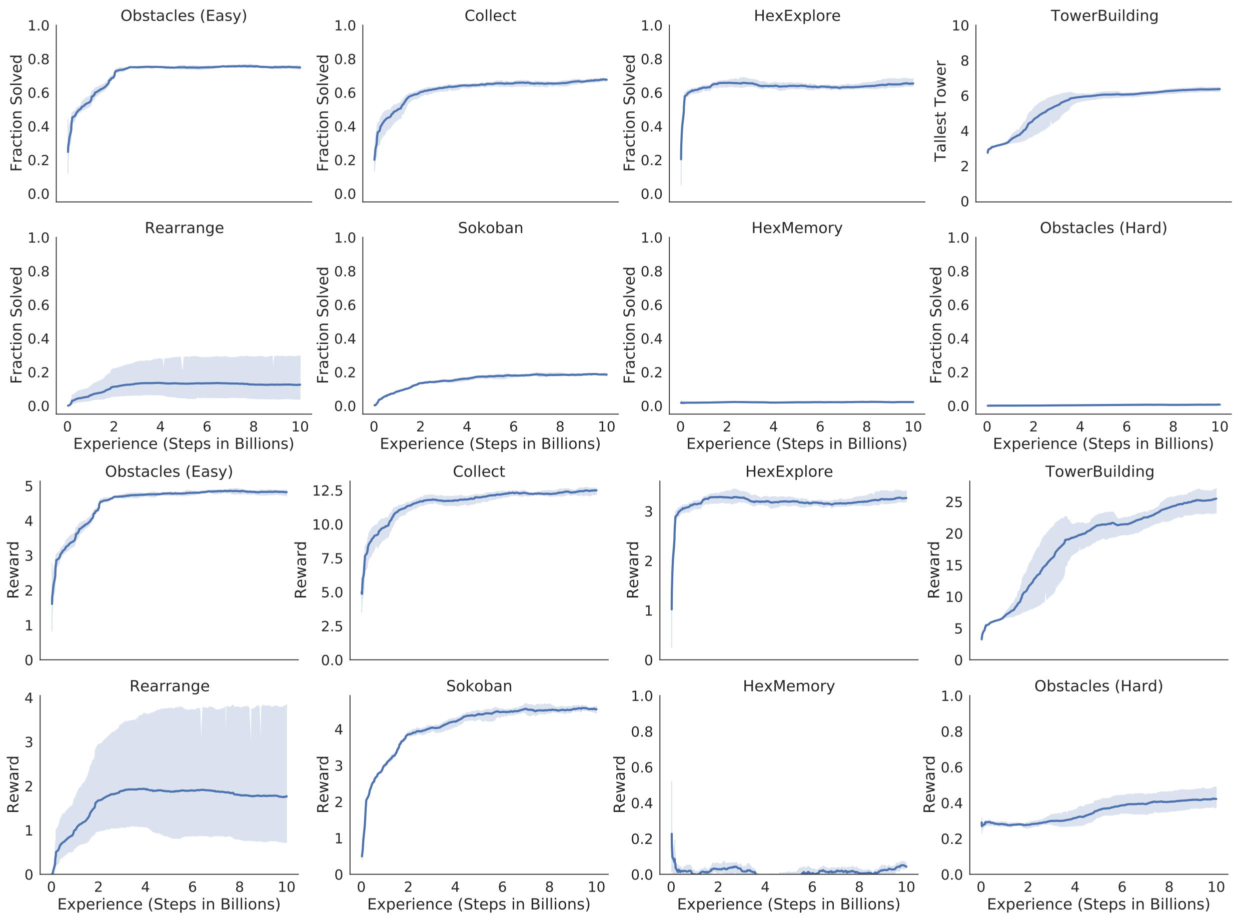

Figure C.3. Single agent with CPC|A training sessions extended to 10^{10} frames. Both task completion (true objective) and episodic rewards are reported. Here the results are averaged over three random seeds.

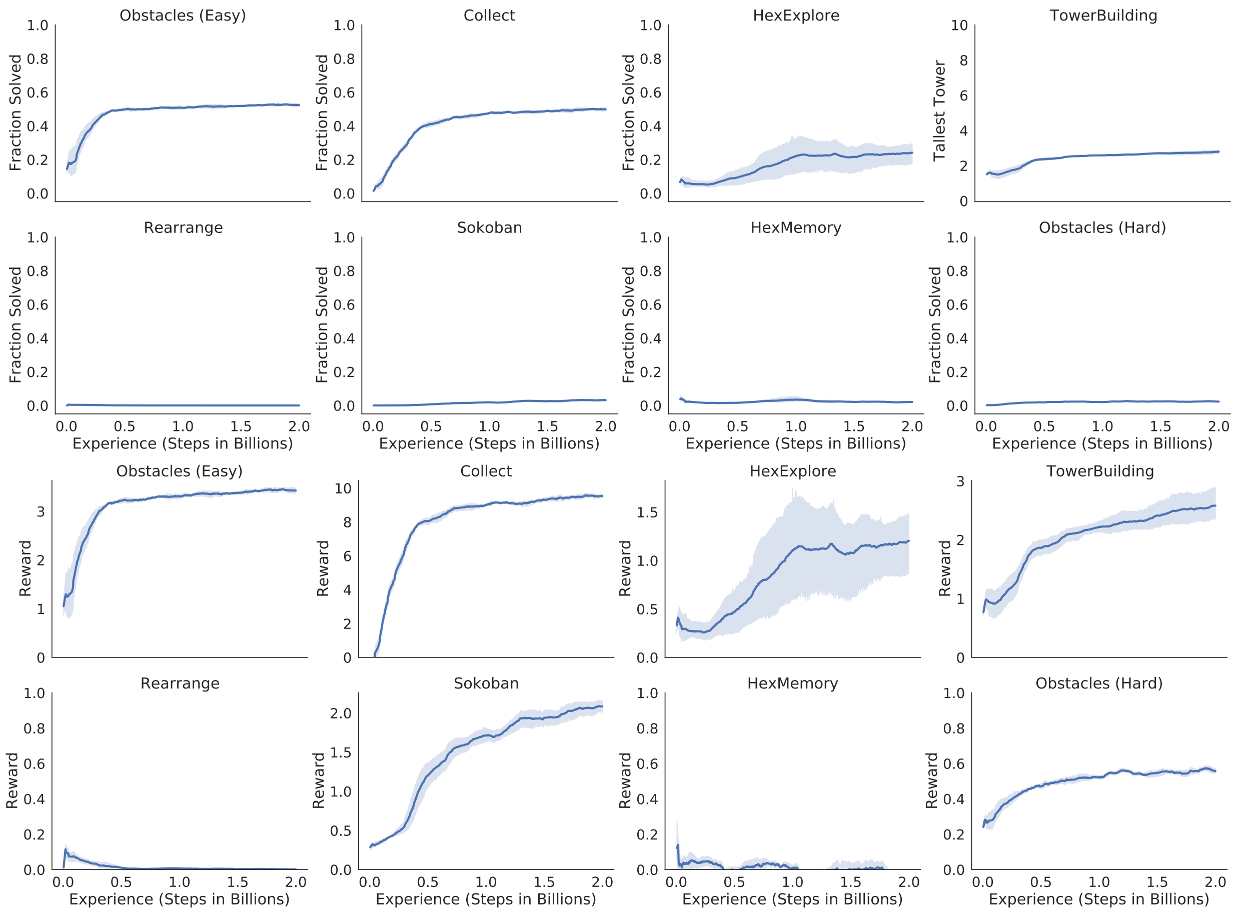

Figure C.4. Performance of a single agent trained on all eight Megaverse-8 tasks. Both task completion (true objective) and episodic rewards are reported. Here the results are averaged over five random seeds.

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *arXiv:1606.01540*, 2016.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Guo, Z. D., Azar, M. G., Piot, B., Pires, B. A., and Munos, R. Neural predictive belief representations. *arXiv:1811.06407*, 2018.

Jakob, W., Rhinelander, J., and Moldovan, D. pybind11 – Seamless operability between C++11 and Python, 2017. https://github.com/pybind/pybind11.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Petrenko, A., Huang, Z., Kumar, T., Sukhatme, G., and Koltun, V. Sample factory: Egocentric 3D control from pixels at 100000 FPS with asynchronous reinforcement learning. In *ICML*, 2020.