VER: Scaling On-Policy RL Leads to the Emergence of Navigation in Embodied Rearrangement Supplementary Materials

Erik Wijmans[1,2] **Irfan Essa**[1,3] **Dhruv Batra**[2,1]
[1] Georgia Institute of Technology [2] Meta AI [3] Google Atlanta
{etw,irfan,dbatra}@gatech.edu

A Systems Considerations

A.1 Memory benefits of not overlapping experience collection and learning

VER does not overlap experience collection and learning. This allows the main process to act as an inference worker during experience collection and a learner during learning, saving 1 CUDA context ($\sim$1GB of GPU memory).

Unlike AsyncOnRL, VER is able to use GPU shared memory to transfer experience from inference workers to learner. This is because VER uses (at least) half the GPU memory for storing rollouts. AsyncOnRL must maintain at least 1 set of rollout buffers for storing the rollout currently being used for learning an another set to store the next rollout being collected. In practice, they maintain more than this minimum 2x to enable faster training.

A.2 Graphics Processing Units (GPUs)

The application of graphics processing units (GPUs) to training neural networks has led to many advances in artificial intelligence. In reinforcement learning, GPUs are used for policy inference, learning the policy, and rendering visual observations like Depth or RGB. Given the importance of GPUs in reinforcement learning, there are several important attributes of them to understand.

1) GPUs are not optimized for multiple processes using them concurrently. They are unable to efficiently execute multiple processes at once (if at all). While sometimes concurrent use is the only option, a system should whenever possible have a single process that has exclusive use of the GPU and uses it fully. 2) GPU drivers have different compute modes for graphics (*i.e.* rendering) and compute (*i.e.* neural network inference). Current GPU drivers have to perform an expensive context switch whenever alternating between these modes. Thus a system should minimize the number of times the GPU must switch between compute and graphics. 3) The compute mode CUDA context for process and each CUDA context requires a large amount of memory. Thus a system should use the minimal number of CUDA contexts possible.

B Skill Deployment

We use the same skills and deployement procedure as Szot et al. [2021] with two changes to the navigation skill: 1) We have a dedicated stop action for navigation instead of using $a_t = \mathbf{0}$ as stop. We find this to be more robust. 2) We add a force penalty to training, this makes the navigation skill less likely to bump into things by accident. However, this also makes it stop immediately if it ever starts in contact with the environment. During evaluation, we mask its prediction of stop if the target object is more than 2 meters away. This information is part of its sensor suite, so this does not require any privileged information.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Optimizer	Adam [Kingma and Ba, 2015]
Initial Learning Rate	2.5×10^{-4}
Final Learning Rate	0
Decay Schedule	Cosine
Total Training Steps	500 Million
PPO Epochs	3
Mini-batches per Epoch	2
PPO Clip	0.2
Initial Entropy Coefficient	10^{-3}
Entropy Coefficient Bounds	$[10^{-4}, 1.0]$
Target Entropy	0.0
Value Loss Coefficient	0.5
Clipped Value Loss	No
Normalized Advantage	No
GAE Parameter [Schulman et al., 2016] (λ)	0.95
Discount Factor (γ)	0.99
VER Important Sampling	Yes
Max VER IS	1.0
Number of Environments (N) – per GPU	16
Experience per rollout ($T \times N$) – per GPU	128×16
Number of GPUs	8

Table A1: Hyperparameters

On Set Table, we add an additional stage to the planner that calls navigation again after open cabinet as the open cabinet skill with base movement tends to move away from the cabinet.

C Videos

In the supplement, we include the following videos.

Video 1 (1-tp-srl-no-nav.mp4) This is an example of TP-SRL(NoNav)+All Base Movement on Tidy House. The objects to pick are have a white wireframe box drawn around them and at their place locations there is another copy of that object. The grey sphere on the ground denotes where the navigation skill was trained to navigate to. This information is not shown to the policy.

The policy makes errors and picks up the wrong object

Video 2 (2-tp-srl.mp4) This is an example of TP-SRL+All Base Movement on Tidy House.

Video 3 (3-pick-*.mp4) Examples of the pick policy on its training task. Note the difference in spawn distance between this and Video 1.

Video 3 (3-place-*.mp4) Examples of the place policy on its training task.

D Hyperparamaters

For CPC|A, we use the hyperparameters from Guo et al. [2020].

The hyperparameters for our rearrangement skills are in Table A1.

For the benchmark comparisons, we set the rollout length T to 128 for methods that require this, that way all methods collect the same amount of experience per rollout. For SampleFactory, we use a batch size of 1024 on 1 GPU, 2048 on 2, and 4096 on 4 and 8 (this the GPU memory limit). We do no use a learned entropy coefficient for benchmarking as not all frameworks support this. We use a fixed value of 10^{-4} as this works well for Open Fridge.

RNN	HTS-RL (Provided)	HTS-RL (Ours)	NoVER	VER
×	242	506	501	620
✓	-	450	462	590

Table A2: **HTS-RL comparison.** Mean system throughput (SPS) over 1 million training steps. HTS-RL (Provided) does not support training recurrent policies. Hardware: Nvidia 2080 Ti with 16 CPUs.

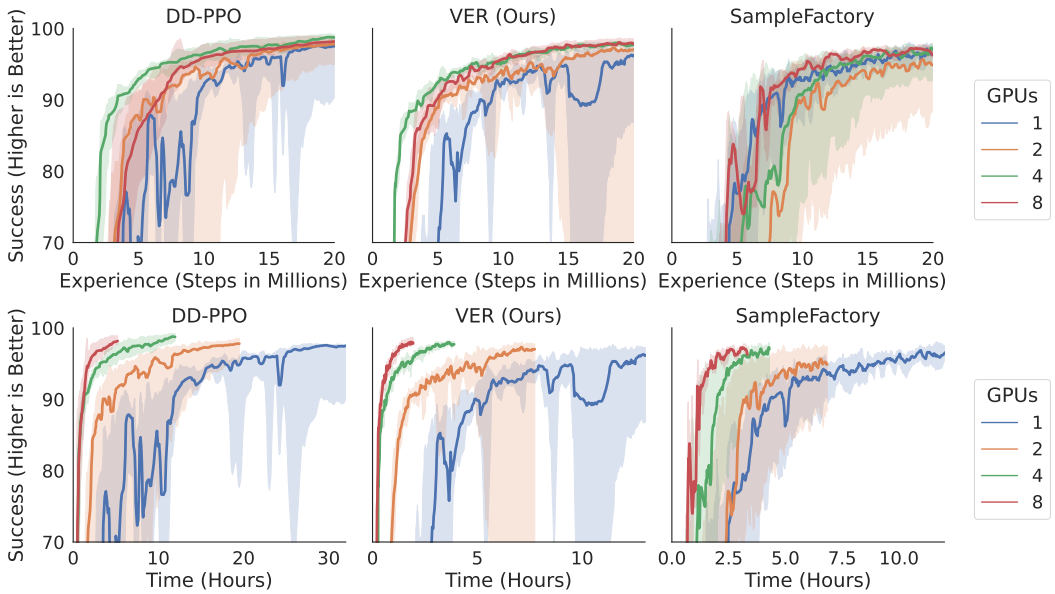

Figure A1: Sample efficiency and time-to-sample for each system on a varying number of GPUs. Even with the reduction of sample efficiency when increasing the number of GPUs, increasing the the number of GPUs always reduces the time to reach convergence.

E HTS-RL Comparison

HTS-RL [Liu et al., 2020] uses the sampling method of AsyncOnRL to collect experience for SyncOnRL (the same as NoVER and VER). It further uses delayed gradients to enable overlapped experience and learning. Unfortunately the provided implementation[1] has inefficiencies that limit its throughput.

To demonstrate this, we add the same style of overlapped experience collection and learning to NoVER and compare the provided HTS-RL implementation, HTS-RL (Provided), with our re-implementation, HTS-RL (Ours). Our implementation 110% faster than the provided implementation (Table A2). Key differences in our implementation are fast userspace mutexes (futexes) in shared memory for synchronization (vs. spin locks), pre-allocated pinned memory for CPU to GPU transfers (vs. allocating for each transfer), and GPU shared memory to send experience from inference workers to learn and weights from learner to inference workers (vs. CPU shared memory).

We also find that given the efficiency of our implementation, there is no significant change in system SPS if we remove overlapped experience collection and learning (Table A2, HTS-RL (Ours) vs. NoVER) for Habitat-style tasks. Removing this reduces GPU memory usage (∼2 GB in our experiments, this will be larger for larger networks). We note that overlapped experience collection and learning does have uses, *i.e.* for CPU simulation or policies with significant CPU components, but it isn't necessary when both the policy and simulator make heavy use of the GPU.

[1] https://github.com/IouJenLiu/HTS-RL

F Datasets

ReplicaCAD [Szot et al., 2021] – Creative Commons Attribution 4.0 International (CC BY 4.0) license. This dataset was artist constructed.

YCB dataset [Calli et al., 2015] – Creative Commons Attribution 4.0 International (CC BY 4.0). This dataset contains 3D scans of generic objects. There is no PII.

Matterport 3D [Chang et al., 2017] – http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf. Consent was given by the owners of the space.

Habitat Matterport Research Dataset [Ramakrishnan et al., 2021] – https://matterport.com/matterport-end-user-license-agreement-academic-use-model-data. Consent was given by the owners of the space and any PII was blurred.

References

Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015. 4

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *International Conference on 3D Vision (3DV)*, 2017. License: http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf. 4

Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3875–3886. PMLR, 2020. 2

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 2

Iou-Jen Liu, Raymond Yeh, and Alexander Schwing. High-throughput synchronous deep rl. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020. 3

Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *Neural Information Processing Systems – Benchmarks and Datasets*, 2021. 4

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 2

Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 4