SECANT: Supplementary Material

Linxi Fan [1 2] **Guanzhi Wang** [1] **De-An Huang** [2] **Zhiding Yu** [2]
Li Fei-Fei [1] **Yuke Zhu** [3 2] **Anima Anandkumar** [4 2]

1. Algorithm Details

In this section, we provide a detailed account of our algorithm implementation and hyperparameters.

We implement Soft Actor Critic (Haarnoja et al., 2018a;b) for all environments in PyTorch v1.7 (Paszke et al., 2019) with GPU acceleration for visual observations. We follow the random cropping scheme in Kostrikov et al. (2020), which first applies a reflection padding to expand the image, and then crop back to the original size. Some of the augmentation operators are implemented with the Kornia library (Riba et al., 2020).

All training hyperparameters are listed in Table 1. We perform a small grid search for the learning rates, and tune them only on the training environment. All agents are optimized by Adam (Kingma & Ba, 2015) with the default settings ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$) in PyTorch. We plan to open-source both our training code and the newly introduced benchmark to facilitate future research in zero-shot policy generalization.

1.1. Inference Latency

At inference time, latency between observing and acting is crucial for real-world deployment. Unlike SECANT that only performs a single forward pass, PAD (Hansen et al., 2020) requires expensive test-time gradient computation. In addition, PAD needs a deeper ConvNet as encoder and extra test-time image augmentations for the auxiliary self-supervised mechanism to work well (Hansen et al., 2020). These add even more overhead during inference.

We expand on Section 5.3 in the main paper, and benchmark both SECANT and PAD on actual hardware for DMControl and CARLA (Fig. 1). The CPU model is Intel Xeon Gold 5220 (2.2 GHz) CPU, and the GPU model is Nvidia RTX 2080Ti. The latency is averaged over 1000 inference steps and excludes the simulation time. We show that SECANT improves inference speed by an order of magnitude in both environments.

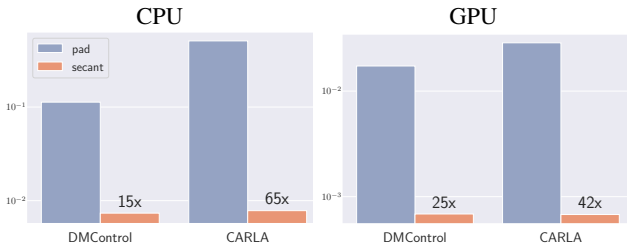

Figure 1. SECANT vs PAD inference latency. Y-axis denotes average seconds per action (log-scale). SECANT improves inference speed by an order of magnitude compared to PAD.

2. Environment Details

2.1. Deepmind Control Suite

We follow the environment settings introduced in Hansen et al. (2020). We use 8 tasks that support randomized colors and 7 tasks that support distracting video background (`Reacher-easy` does not support inserting videos). We use the same action repeat settings as Hansen et al. (2020): 2 for `Finger-spin`, 8 for `Cartpole-swingup` and `Cartpole-balance`, and 4 for the rest. Please refer to Tassa et al. (2018) and Hansen et al. (2020) for more details about DMControl tasks.

2.2. Robosuite

We use the Franka Panda robot model with operational space control for 4 Robosuite tasks. The action dimensions for door opening, nut assembly, peg-in-hole, and two-arm lifting are 7, 7, 12 and 14, respectively. We use continuous action space. The control frequency is set to 20Hz, which means that the robot receives 20 control signals during every simulation second. We provide brief descriptions of each task and their associated reward functions below. All environments add an extra positive reward upon task completion, in addition to the dense reward shaping. Example observations are shown in Fig. 2. Please refer to Zhu et al. (2020) for more details.

Door opening. A robot arm must learn to turn the handle and open the door in front of it. The reward is shaped by the distance between the door handle and the robot arm, and the rotation angle of the door handle.

Table 1. SECANT hyperparameters for all environments.

Hyperparameter	DMControl	Robosuite	CARLA	iGibson
Input dimension	$9 \times 84 \times 84$	$9 \times 168 \times 168$	$9 \times 84 \times 420$	$9 \times 168 \times 168$
Stacked frames	3	3	3	3
Discount factor γ	0.99	0.99	0.99	0.99
Episode length	1000	500	1000	500
Number of training steps	500K	800K	500K	800K
SAC replay buffer size	100K	100K	100K	50K
SAC batch size	512	512	1024	128
Optimizer	Adam	Adam	Adam	Adam
Actor learning rate	5e-4 (Walker-walk) 1e-3 (otherwise)	1e-4 (Peg-in-hole) 1e-3 (otherwise)	1e-3	5e-4
Critic learning rate	5e-4 (Walker-walk) 1e-3 (otherwise)	1e-4	1e-3	1e-4
$\log \alpha$ learning rate	5e-4 (Walker-walk) 1e-3 (otherwise)	1e-4 (Peg-in-hole) 1e-3 (otherwise)	1e-3	5e-4
Critic target update frequency	2	4	2	4
Random cropping padding	4	8	(4, 12)	8
Encoder conv layers	4	4	4	4
Encoder conv strides	[2, 1, 1, 2]	[2, 1, 1, 2]	[2, (1, 2), (1, 2), 2]	[2, 2, 1, 1]
Encoder conv channels	32	32	[64, 64, 64, 32]	32
Encoder feature dim	50	50	64	50
Actor head MLP layers	3	3	3	3
Actor head MLP hidden dim	1024	1024	1024	1024
SECANT student augmentation	Combo1	Combo2	Combo1	Combo2
SECANT learning rate	1e-3	1e-3	1e-3	1e-3
SECANT replay buffer size	10K	20K	10K	20K
SECANT batch size	512	512	1024	512

Nut assembly. Two colored pegs (one square and one round) are mounted on the tabletop. The robot must fit the round nut onto the round peg. At first, the robot receives a reaching reward inversely proportional to the distance between the gripper and the nut, and a binary reward once it grasps the nut successfully. After grasping, it obtains a lifting reward proportional to the height of the nut, and a hovering reward inversely proportional to the distance between the nut and the round peg.

Peg-In-Hole. One arm holds a board with a square hole in the center, and the other holds a long peg. The two arms must coordinate to insert the peg into the hole. The reward is shaped by the distance between two arms, along with the orientation and distance between the peg and the hole.

Two-arm lifting. A large pot with two handles is placed on the table. Two arms on opposite ends must each grab a handle and lift the pot together above certain height, while keeping it level. At first, the agent obtains a reaching reward inversely proportional to the distance between each arm and its respective pot handle, and a binary reward if each gripper is grasping the correct handle. After grasping, the agent receives a lifting reward proportional to the pot's height above the table and capped at a certain threshold.

No agent can solve nut assembly and two-arm lifting completely. However, SECANT is able to obtain partial rewards by grasping the nut or the pot handles successfully in the unseen test environments, while the prior SOTA methods struggle. There is still room to improve on this challenging benchmark.

2.3. CARLA

For autonomous driving in CARLA, the goal of the agent is to drive as far as possible on an 8-figure highway without collision under diverse weather conditions. We implement the environment in CARLA v0.9.9.4 (Dosovitskiy et al., 2017) and adopt the reward function in Zhang et al. (2020):

$$r_t = \mathbf{v}_{\text{agent}}^\top \hat{\mathbf{u}}_{\text{highway}} \cdot \Delta t - \lambda_c \cdot \text{collision} - \lambda_s \cdot |\text{steer}|$$

where $\mathbf{v}_{\text{agent}}$ is the velocity vector of our vehicle, and the dot product with the highway's unit vector $\hat{\mathbf{u}}_{\text{highway}}$ encourages progression along the highway as fast as possible. $\Delta t = 0.05$ discretizes the simulation time. We penalize collision, measured as impulse in Newton-seconds, and excessive steering. The respective coefficients are $\lambda_c = 10^{-4}$ and $\lambda_s = 1$. We do not investigate more sophisticated rewards like lane-keeping and traffic sign compliance, as they are not the main focus of this paper. We use action repeat 8 for all agents.

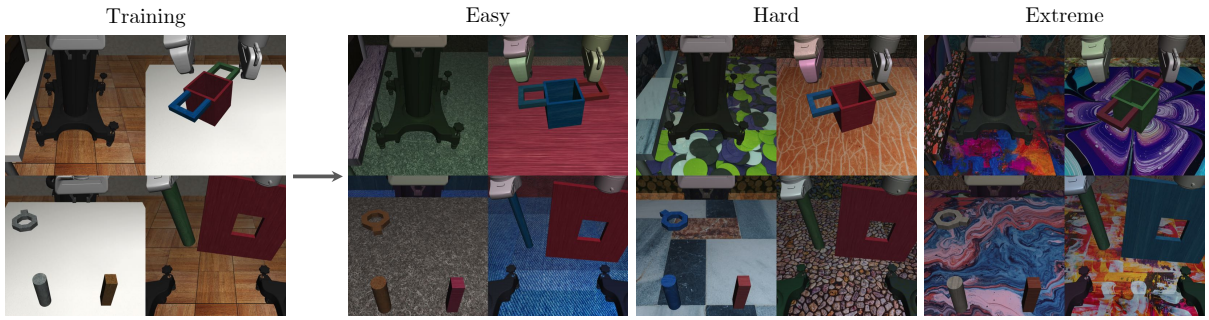

Figure 2. Sample Robosuite environments. Tasks in clockwise order: Door opening, Two-arm lifting, Peg-in-hole, and Nut assembly.

Table 2. iGibson object navigation: detailed breakdown of *Easy* and *Hard* settings.

Setting	Room	SECANT (Ours)	SAC	SAC+crop	DR	NetRand	SAC+IDM	PAD
Easy Rooms	Beechwood	**63.0 ± 17.5**	17.0 ± 10.4	16.0 ± 8.2	21.0 ± 10.2	47.0 ± 7.6	28.0 ± 8.4	33.0 ± 13.0
	Ihlen	**58.0 ± 19.2**	9.0 ± 4.2	11.0 ± 8.2	6.0 ± 5.5	42.0 ± 9.1	24.0 ± 10.8	25.0 ± 7.9
	Merom	**65.0 ± 14.6**	20.0 ± 7.1	15.0 ± 6.1	20.0 ± 12.7	38.0 ± 9.7	24.0 ± 6.5	26.0 ± 6.5
	Wainscott	**64.0 ± 8.2**	18.0 ± 2.7	15.0 ± 7.1	41.0 ± 15.2	49.0 ± 14.7	35.0 ± 10.6	29.0 ± 12.4
	Benevolence-0	**67.0 ± 15.7**	15.0 ± 3.5	16.0 ± 9.6	17.0 ± 8.4	49.0 ± 8.2	39.0 ± 11.4	51.0 ± 8.2
	Benevolence-1	**48.0 ± 10.4**	6.0 ± 6.5	13.0 ± 5.7	6.0 ± 6.5	34.0 ± 8.2	18.0 ± 10.4	21.0 ± 8.2
	Benevolence-2	**59.0 ± 22.2**	14.0 ± 8.2	8.0 ± 6.7	21.0 ± 11.9	38.0 ± 9.7	31.0 ± 17.1	42.0 ± 11.5
	Pomaria-1	**47.0 ± 24.6**	13.0 ± 9.1	10.0 ± 7.1	15.0 ± 7.9	27.0 ± 10.4	22.0 ± 16.0	34.0 ± 7.4
	Pomaria-2	**58.0 ± 14.4**	16.0 ± 6.5	15.0 ± 7.9	14.0 ± 10.2	32.0 ± 9.1	22.0 ± 9.1	27.0 ± 4.5
	Rs	**39.0 ± 8.2**	10.0 ± 6.1	10.0 ± 5.0	15.0 ± 9.4	36.0 ± 13.4	16.0 ± 8.2	21.0 ± 9.6
Hard Rooms	Beechwood	**61.0 ± 12.4**	7.0 ± 5.7	5.0 ± 5.0	10.0 ± 3.5	31.0 ± 9.6	7.0 ± 2.7	16.0 ± 8.2
	Ihlen	**37.0 ± 14.0**	4.0 ± 4.2	10.0 ± 5.0	6.0 ± 5.5	30.0 ± 5.0	9.0 ± 8.2	26.0 ± 9.6
	Merom	**45.0 ± 5.0**	4.0 ± 4.2	4.0 ± 4.2	10.0 ± 11.7	32.0 ± 10.4	11.0 ± 4.2	25.0 ± 9.4
	Wainscott	**46.0 ± 9.6**	10.0 ± 7.1	10.0 ± 3.5	10.0 ± 5.0	23.0 ± 4.5	12.0 ± 5.7	9.0 ± 6.5
	Benevolence-0	**56.0 ± 7.4**	7.0 ± 2.7	6.0 ± 2.2	9.0 ± 6.5	36.0 ± 8.2	11.0 ± 2.2	11.0 ± 8.9
	Benevolence-1	44.0 ± 4.2	12.0 ± 9.7	8.0 ± 5.7	**49.0 ± 14.3**	36.0 ± 8.9	13.0 ± 10.4	15.0 ± 12.7
	Benevolence-2	**49.0 ± 15.6**	21.0 ± 10.8	12.0 ± 7.6	31.0 ± 10.8	44.0 ± 8.9	28.0 ± 7.6	36.0 ± 2.2
	Pomaria-1	**41.0 ± 2.2**	9.0 ± 6.5	12.0 ± 2.7	11.0 ± 8.2	26.0 ± 4.2	10.0 ± 6.1	20.0 ± 12.2
	Pomaria-2	**57.0 ± 5.7**	13.0 ± 5.7	6.0 ± 4.2	8.0 ± 6.7	54.0 ± 12.9	18.0 ± 7.6	17.0 ± 10.4
	Rs	41.0 ± 5.5	6.0 ± 2.2	6.0 ± 6.5	8.0 ± 6.7	26.0 ± 7.4	8.0 ± 4.5	**86.0 ± 2.2**

2.4. iGibson

The goal of the agent in iGibson (Xia et al., 2020; Shen et al., 2020) is to find a lamp hanging from the ceiling and navigate to it as closely as possible. Our agent is a virtual LoCoBot (Gupta et al., 2018). The action dimension is 2, which controls linear velocity and angular velocity. We use continuous action space with 10Hz control frequency.

Table 6 of the main paper reports the average of 10 *Easy* and 10 *Hard* rooms. We provide a detailed breakdown of those results in Table 2 over different floorplans. The *Easy* and *Hard* settings feature distinct interior decorations with different visual distribution shifts from the training room.

3. Additional Ablation Studies

3.1. More Augmentation Strategies

In addition to the ablations in Section 5.1, we present extensive results with alternative 2-stage and 1-stage augmentation strategies in Table 3. The columns "No-aug → Weak" and "No-aug → Strong" are student distillation with weak and strong augmentations, respectively. "No-aug" column denotes the single-stage policy trained with no augmentation and directly evaluated on unseen tests. "Strong-only" column is a single-stage policy trained with Combo1 (Section 4.2) augmentation. SECANT outperforms these baselines in a variety of Robosuite and DMControl tasks, which demonstrates that weakly-augmented expert followed by strongly-augmented student is indeed necessary for achieving SOTA performance.

3.2. SECANT-Parallel Results on Robosuite

We include more experiments with the SECANT-Parallel variant on Robosuite (table 4) in addition to the DMControl results in Section 5.1 of the main paper. The performance numbers further validate that it is beneficial to train the expert and the student in sequence, rather than in parallel.

Table 3. Additional ablation studies on alternative augmentation strategies.

Setting	Task	SECANT	No-aug	No-aug → Weak	No-aug → Strong	Strong-only
Robosuite *Easy*	Door opening	**782 ± 93**	17 ± 12	37 ± 21	367 ± 130	47 ± 52
	Nut assembly	**419 ± 63**	3 ± 2	8 ± 1	172 ± 83	143 ± 95
	Two-arm lifting	**610 ± 28**	29 ± 11	43 ± 15	100 ± 8	93 ± 26
	Peg-in-hole	**837 ± 42**	186 ± 62	185 ± 67	489 ± 32	287 ± 63
Robosuite *Hard*	Door opening	**522 ± 131**	11 ± 10	31 ± 15	270 ± 94	36 ± 37
	Nut assembly	**437 ± 102**	6 ± 7	13 ± 5	150 ± 40	136 ± 32
	Two-arm lifting	**624 ± 40**	28 ± 11	46 ± 12	99 ± 9	101 ± 37
	Peg-in-hole	**774 ± 76**	204 ± 81	201 ± 53	353 ± 74	290 ± 81
Robosuite *Extreme*	Door opening	**309 ± 147**	11 ± 10	24 ± 15	190 ± 37	32 ± 31
	Nut assembly	**138 ± 56**	2 ± 1	5 ± 3	33 ± 11	43 ± 28
	Two-arm lifting	**377 ± 37**	25 ± 7	46 ± 12	65 ± 13	58 ± 27
	Peg-in-hole	**520 ± 47**	164 ± 63	197 ± 66	285 ± 80	289 ± 66
DMControl *Color*	Cheetah run	**582 ± 64**	133 ± 26	76 ± 23	160 ± 29	296 ± 13
	Ball in cup catch	**958 ± 7**	151 ± 36	125 ± 26	161 ± 17	777 ± 51
	Cartpole swingup	**866 ± 15**	248 ± 24	231 ± 31	296 ± 27	628 ± 118
	Walker walk	**856 ± 31**	144 ± 19	81 ± 13	153 ± 16	598 ± 47
DMControl *Video*	Cheetah run	**428 ± 70**	80 ± 19	63 ± 15	158 ± 30	271 ± 20
	Ball in cup catch	**903 ± 49**	172 ± 46	134 ± 33	143 ± 8	727 ± 59
	Cartpole swingup	**752 ± 38**	204 ± 20	245 ± 17	285 ± 29	503 ± 99
	Walker walk	**842 ± 47**	104 ± 14	85 ± 11	148 ± 15	547 ± 51

Table 4. SECANT-Parallel variant on Robosuite. It is advantageous to train expert and student sequentially rather than in parallel.

Setting	Task	SECANT	SECANT-Parallel
Robosuite Easy	Door opening	**782 ± 93**	529 ± 145
	Nut assembly	**419 ± 63**	374 ± 64
	Two-arm lifting	**610 ± 28**	390 ± 83
	Peg-in-hole	**837 ± 42**	540 ± 80
Robosuite Hard	Door opening	**522 ± 131**	399 ± 71
	Nut assembly	**437 ± 102**	429 ± 80
	Two-arm lifting	**624 ± 40**	348 ± 89
	Peg-in-hole	**774 ± 76**	598 ± 123
Robosuite Extreme	Door opening	309 ± 147	**335 ± 125**
	Nut assembly	**138 ± 56**	104 ± 56
	Two-arm lifting	**377 ± 37**	114 ± 30
	Peg-in-hole	**520 ± 47**	382 ± 161

References

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

Gupta, A., Murali, A., Gandhi, D. P., and Pinto, L. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in Neural Information Processing Systems*, 31:9094–9104, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications, 2018b.

Hansen, N., Sun, Y., Abbeel, P., Efros, A. A., Pinto, L., and Wang, X. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Riba, E., Mishkin, D., Ponsa, D., Rublee, E., and Bradski, G. Kornia: an open source differentiable computer vision library for pytorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3674–3683, 2020.

Shen, B., Xia, F., Li, C., Martın-Martın, R., Fan, L., Wang, G., Buch, S., D'Arpino, C., Srivastava, S., Tchapmi, L. P., Vainio, K., Fei-Fei, L., and Savarese, S. igibson, a simulation environment for interactive tasks in large realistic scenes. *arXiv preprint*, 2020.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. DeepMind control suite. Technical report, DeepMind, January 2018.

Xia, F., Shen, W. B., Li, C., Kasimbeg, P., Tchapmi, M. E., Toshev, A., Martín-Martín, R., and Savarese, S. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.

Zhang, A., McAllister, R., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Zhu, Y., Wong, J., Mandlekar, A., and Martín-Martín, R. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.