APPENDIX A
MORE IMPLEMENTATION DETAILS

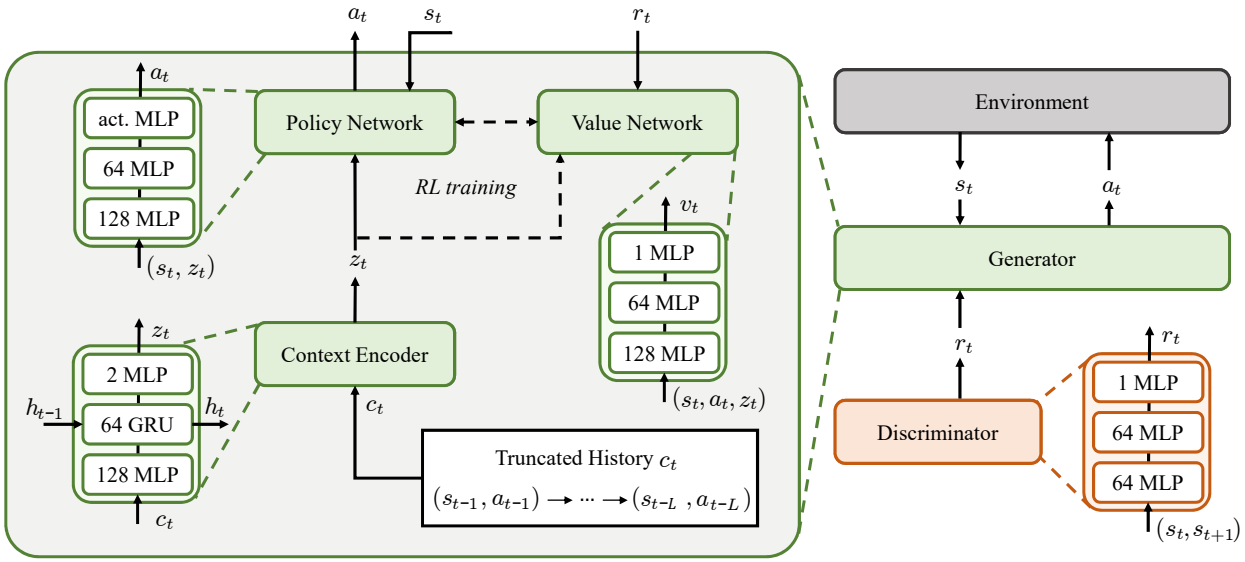

Fig. 1. Overall network structure of GMAIL. The `act.` in the policy network indicates the dimension of the action space.

TABLE 1
Full Hyper-parameters Settings of GMAIL.

	Hyper-parameters	Value
Network	Learning rate of the context encoder	3×10^{-4}
	Learning rate of the policy network	3×10^{-4}
	Learning rate of the value network	1×10^{-3}
	Learning rate of the discriminator	1×10^{-3}
	Batch size of the generator	256
	Batch size of the discriminator	32
	Optimizer	Adam
SAC	Target network update rate τ	0.995
	Discount factor γ	0.99
	Temperature parameter	auto [1] for HalfCheetah and Highway 0.2 for Hopper and Ant
GMAIL	H	4
	History length L	16 for HalfCheetah and Ant 2 for Hopper and Highway
	Absorting state	False for HalfCheetah and Highway True for Hopper and Ant
	Gradient penalty weight	0 for HalfCheetah and Highway 10 for Hopper and Ant
	Total training steps	1×10^6

A.1 Software

We use the following software versions:

- Python 3.8
- MuJoCo 2.2.0 [2]
- Gym 0.21.0 [3]
- MuJoCo-py 2.1.2.14
- PyTorch 1.12.1 [4]

A.2 Hardware

We use the following hardware:

- NVIDIA RTX A4000
- 12th Gen Intel(R) Core(TM) i9-12900K

A.3 Expert Demonstration

Table 2 illustrates returns of the collected expert trajectories and the corresponding ω, where `gravity` is the changeable parameter. For the Highway task, all the returns are 70.

TABLE 2
Return of each collected expert trajectory and the corresponding ω.

ω	HalfCheetah	Hopper	Ant
-2.9968	7002	2621	2002
-2.8878	9445	2369	1960
-2.8332	7001	2685	2195
-2.4872	7858	2818	2840
-2.4434	7517	2416	2737
-2.3782	7005	2543	2603
-1.2358	9310	3543	4291
-1.1043	9999	3778	3570
-0.6169	9532	3606	3966
0.3547	6978	3382	5471
1.1138	5698	3459	6000
1.1564	6702	3227	5005
1.4915	5454	3545	6239
1.8070	6542	3397	6132
2.2714	6244	2666	6089
2.7498	4921	298	5783

A.4 Baselines

Below are details of the implementation of baselines:

• *BC.* [5], one of the traditional imitation learning algorithms, uses supervised learning methods to fit expert policies. In our implementation, we update the policy network by minimizing the mean squared error between predicted and expert actions. The architecture of the policy network is a three-layer MLP with the number of the two hidden layers' units being 400 and 300, respectively.

• *DAC-S.* DAC [6] utilizes the state-action pair as the discriminator's input and trains the generator using TD3 [7]. To be compatible with our setting and for a fair comparison, we implement a discriminator taking the state-next-state pair as input and use SAC [1] to train the generator.

• *InfoGAIL-H.* InfoGAIL [8] considers multi-modal imitation learning and extends GAIL [9] with a posterior network which approximately maximizes the mutual information between the latent space and trajectories, similar to InfoGAN [10]. However, InfoGAIL does not consider predicting modality (i.e., dynamic in our setting) during evaluation, which is essential for the learned policy to adapt to dynamics change. We thus modify InfoGAIL as follows. First, we change the input of the posterior network from (s, a) to (s, s'). Then, we get k predictions using the posterior network with the latest k state transitions. Finally, we use majority voting [11] on the k predictions to infer the current dynamic. The generator is also trained via SAC [1]. We name this modified version as InfoGAIL-H.

• *OOD-IL.* To address the dynamic mismatch issue, OOD-IL [12] computes the transferability of expert demonstrations collected in environments with different dynamics and samples each demonstration with probability proportional to its transferability. Specifically, it first clusters the demonstrations in an unsupervised way and then trains K discriminators, each for a cluster, to calculate the transferability $t(s_t, s_{t+1})$ for every state-next-state pair (s_t, s_{t+1}) as $t(s_t, s_{t+1}) = \sum_{k=1}^{K} \mathbb{I}[(s_t, s_{t+1}) \in \mathcal{D}_k] D_k(s_t, s_{t+1})$. Here, $\mathcal{D}_k$ is the set of expert trajectories in the k-th cluster. D_k is the trained discriminator for the kth cluster. $\mathbb{I}$ is the indicator function. K is the number of clusters. In our implementation, we use $K = 4$ for HalfCheetah and Ant and $K = 3$ for Hopper and Highway. The architectures of the networks are the same as DAC-S. Other settings are kept default as the authors provided in its implementation[1].

1. https://github.com/EvieQ01/OODIL

• *RIME*. [13], which imitates multiple experts in sampled environment dynamics to enhance the robustness to general variations in environment dynamics. Technically, it minimizes the risk with respect to the JS divergence [14] between the agent's policy and each of the sampled experts. We use the implementation provided by the authors[2].

APPENDIX B
MORE EXPERIMENTAL RESULTS

B.1 Full Visualization Results

Figure 2 and Figure 3 illustrate how GMAIL encodes the environment dynamics within an episode in non-stationary environments and how the contexts relate to different `gravity` values on the 3 MuJoCo tasks. GMAIL differentiates among various dynamics and swiftly identifies the current dynamic. In contrast, InfoGAIL-H performs poorly and cannot stably and accurately infer the real dynamic. Furthermore, The contexts generated by GMAIL have a strong linear correlation with the real `gravity` values, and this relationship is maintained in OOD environments. These results demonstrate that GMAIL is highly sensitive to dynamics change and generalizes well to OOD dynamics.

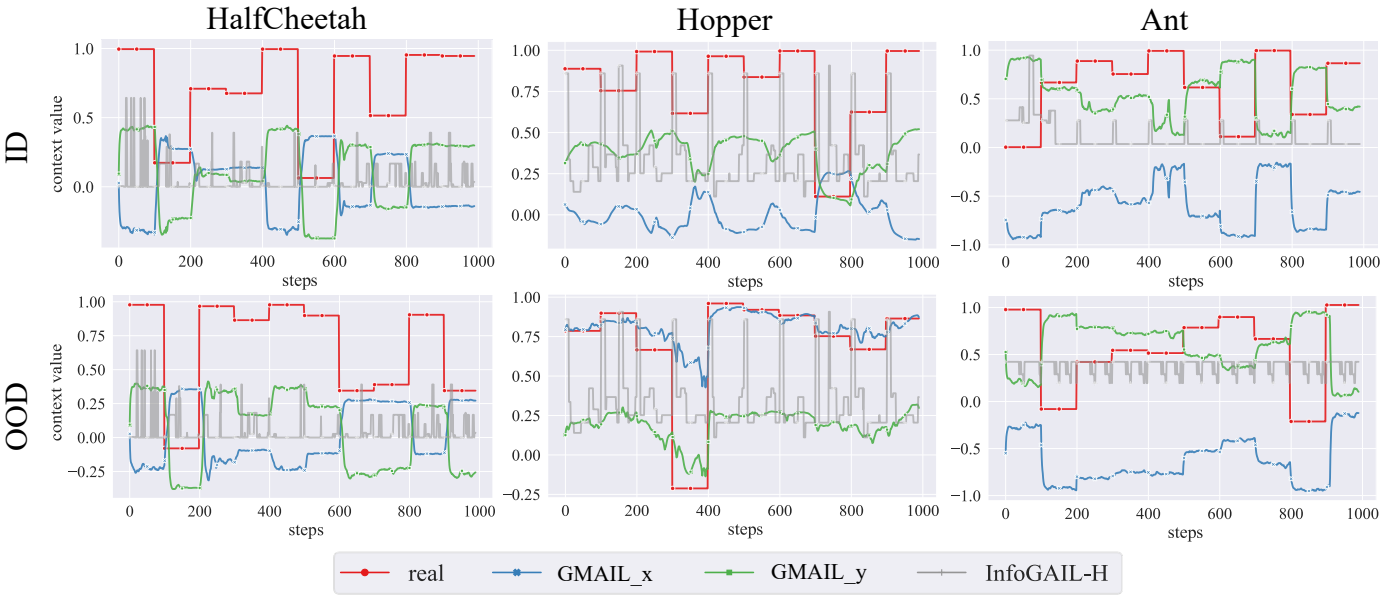

Fig. 2. Latent contexts in ID and OOD non-stationary environments.

B.2 Full Sensitivity Analysis Results

Figure 4 displays the full results of sensitivity analysis experiments on H, where we see that the policy performs well with a properly large H. However, an excessively large H does not help since some sub-optimal states may be mistakenly assigned high rewards.

Figure 5 shows the full results of sensitivity analysis experiments on different history lengths L, which shows that a small or huge L will degrade the performance. We argue that a small L may not provide enough information for the encoder to accurately infer the real `gravity`, whereas a huge L will drag the encoder from sensitively reacting to dynamics change.

Figure 6 shows the full results of sensitivity analysis experiments on different history lengths α, where we can see that a proper α works well across all the environments. We use $\alpha = 3000$ to get the main results as shown in Figure 9. Recall that α controls the scale of the $\|\bar{z}^p - \bar{z}^q\|_2^2$. A big α will make $\exp\left(-\alpha\|\bar{z}^p - \bar{z}^q\|_2^2\right)$ sensitive, while a small α will make it sluggish.

Figure 7 shows the full results of sensitivity analysis experiments on different history lengths λ, where we can see that a proper λ works well across all the environments. We use $\lambda = 50$ to get the main results as shown in Figure 9. Recall that λ controls the weight of historical moving average $\hat{z}^i$. A big λ will hinder the update of the moving average $\hat{z}^i$, while a small λ will make its update unstable.

2. https://github.com/JongseongChae/RIME

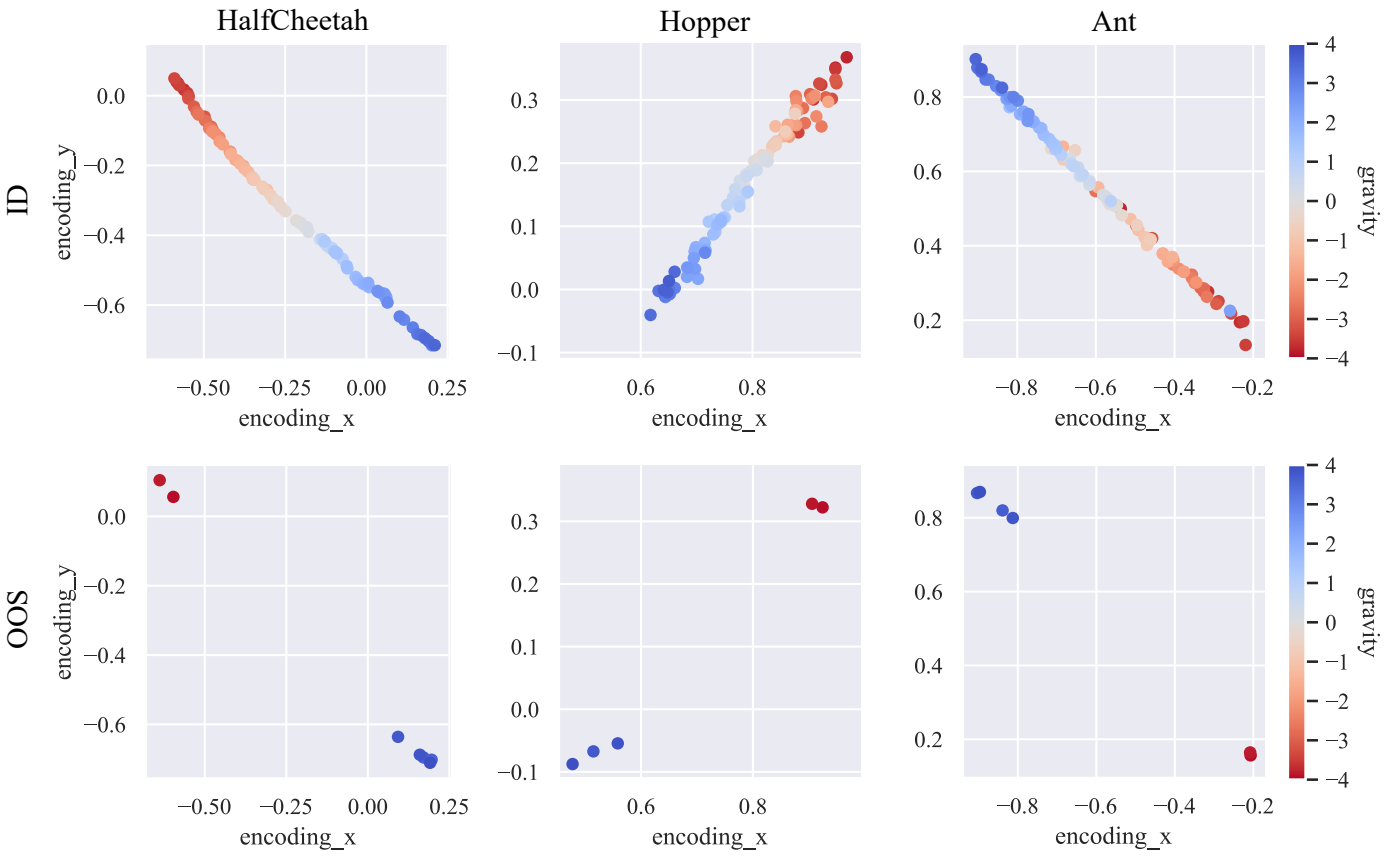

Fig. 3. Visualization of the contexts generated by GMAIL in ID and OOD environments.

B.3 More Results on Other Varying Parameters

Besides `gravity`, we also experiment with another varying parameter, `body mass`, on the three MuJoCo tasks[3]. We change the `body mass` in the same way as `gravity`. Let b_0 denote the default value of `body mass`. Then we get a new environment with its `body mass`

$$b' = 1.5^{\omega} b_0,$$

where ω is a scaling factor. Like the experiments on `gravity`, we first collect expert demonstrations using SAC [1]. Table 3 illustrates the return of each trajectory. For each task, we then train GMAIL and baseline methods for $1M$ steps. Figure 8 presents the results over 3 random seeds, where GMAIL still performs best.

3. We do not experiment on the Highway task because it only has two changeable parameters, `friction` and `mass`, taking effect in an equivalent way.

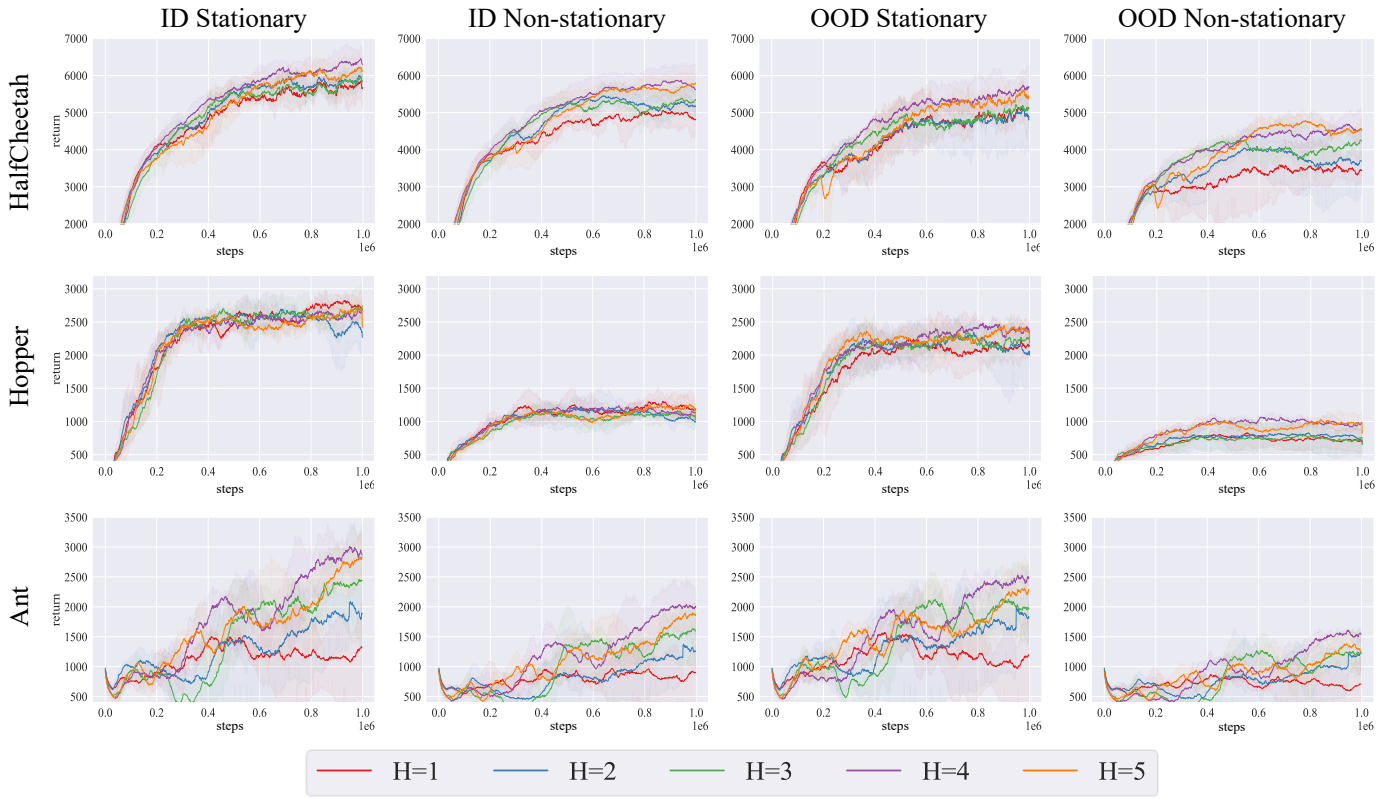

Fig. 4. Full sensitivity analysis results on H.

TABLE 3
Return of each collected trajectory with `body mass` being the varying parameter.

ω	**HalfCheetah**	ω	**Hopper**	ω	**Ant**
-2.6804	10043	-2.8359	3656	-2.8824	4895
-2.2548	5997	-2.7006	3927	-2.4266	5874
-2.1778	10078	-2.3810	3989	-2.0334	5684
-2.1757	9885	-2.2201	4076	-0.8420	6683
-1.5784	10778	-2.1198	3774	-0.4165	6038
-1.2387	6142	-1.8117	3896	-0.3808	5445
-0.9261	8547	-0.6142	3485	-0.0615	5193
-0.8566	2460	-0.0509	3379	0.6354	5906
-0.5154	9902	0.2326	3501	0.6905	5406
-0.4736	8028	0.5190	3225	0.7246	4685
1.0223	6358	1.1510	3234	1.1520	4912
1.1535	6638	1.1983	3260	1.7843	4379
1.3555	9896	1.3557	3149	2.0756	5922
1.5005	5313	1.5006	3048	2.3769	4050
2.0518	10442	2.2580	2872	2.5502	5582
2.7886	1888	2.2686	2875	2.6636	6169

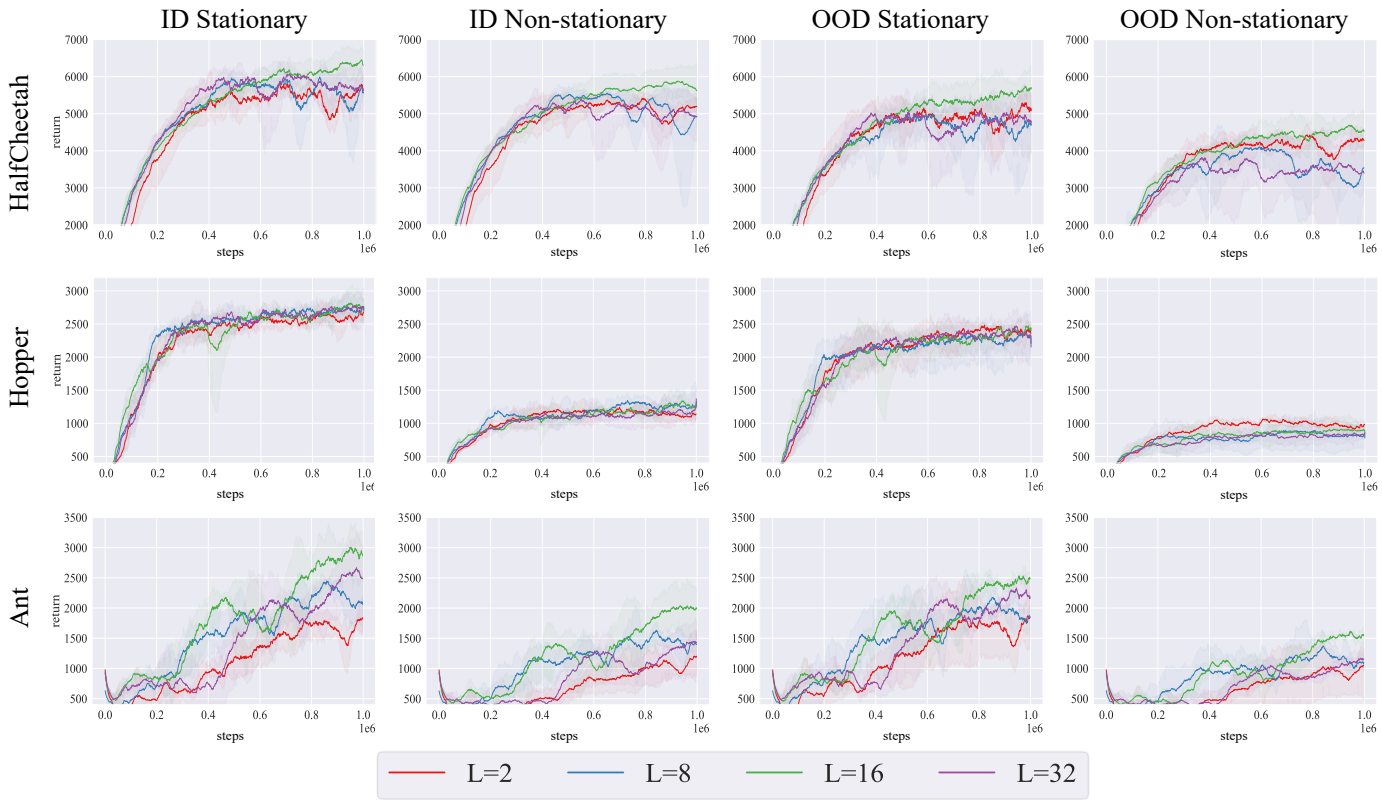

Fig. 5. Full sensitivity analysis results on L.

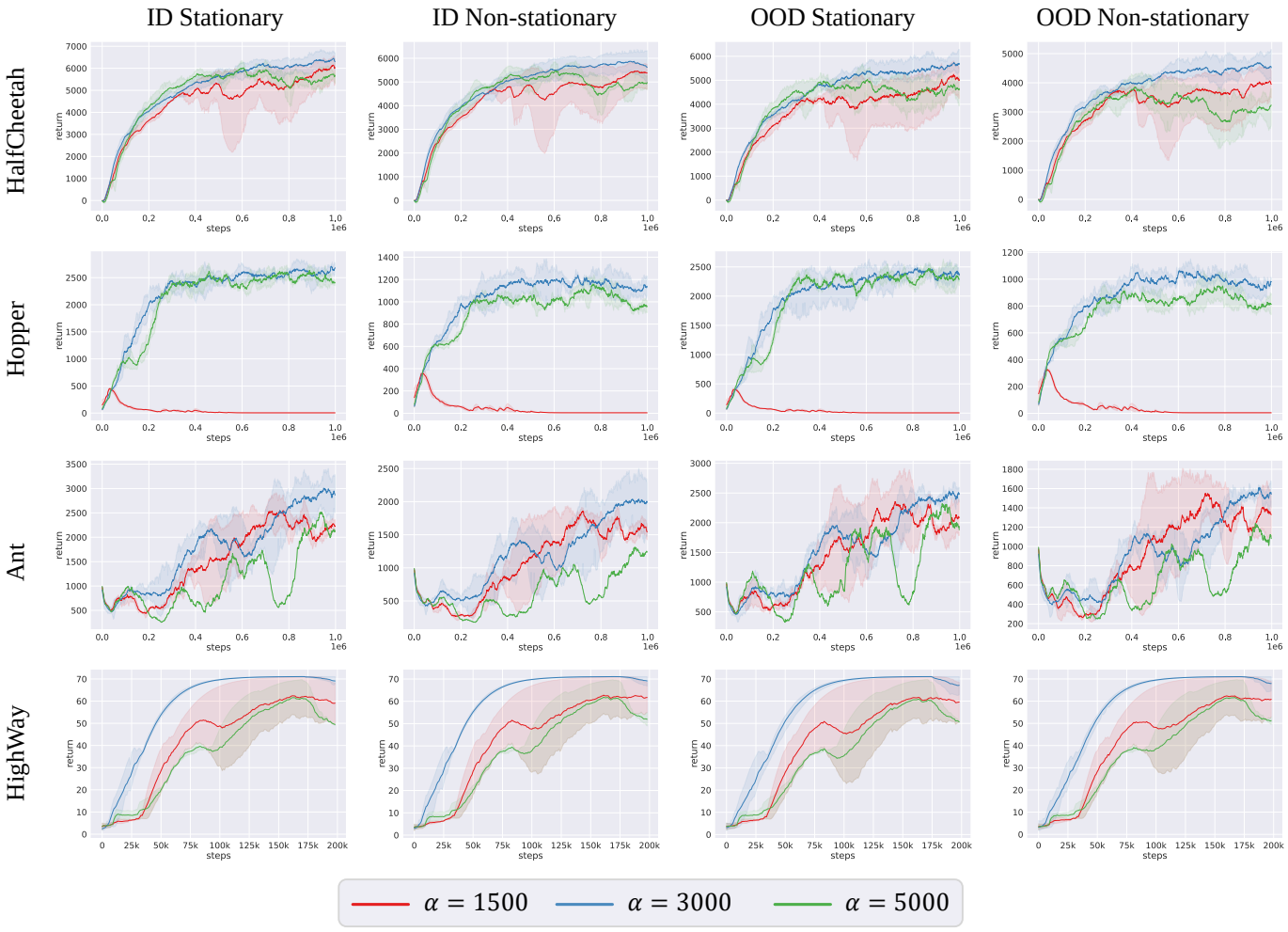

Fig. 6. Full sensitivity analysis results on α.

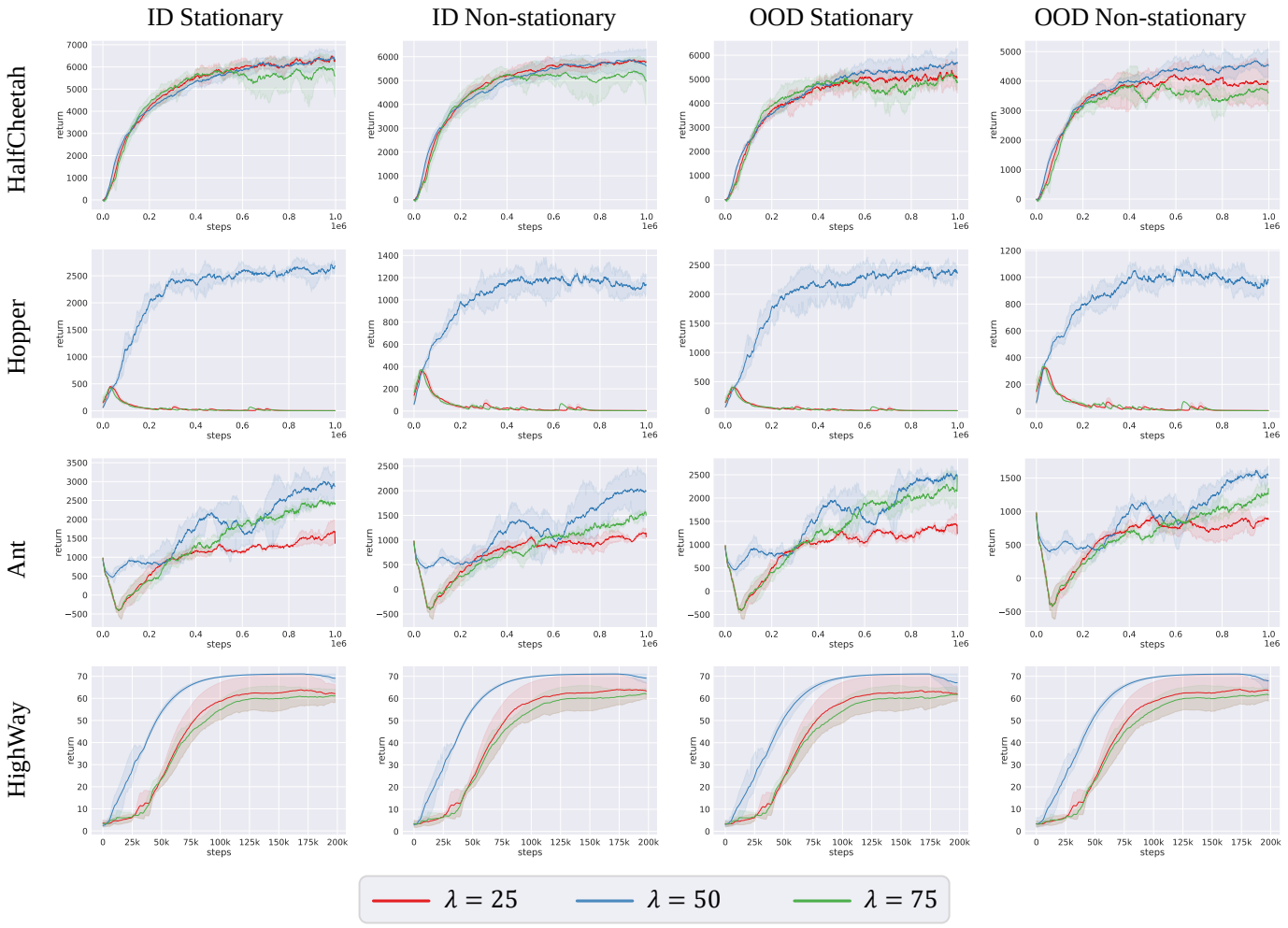

Fig. 7. Full sensitivity analysis results on λ.

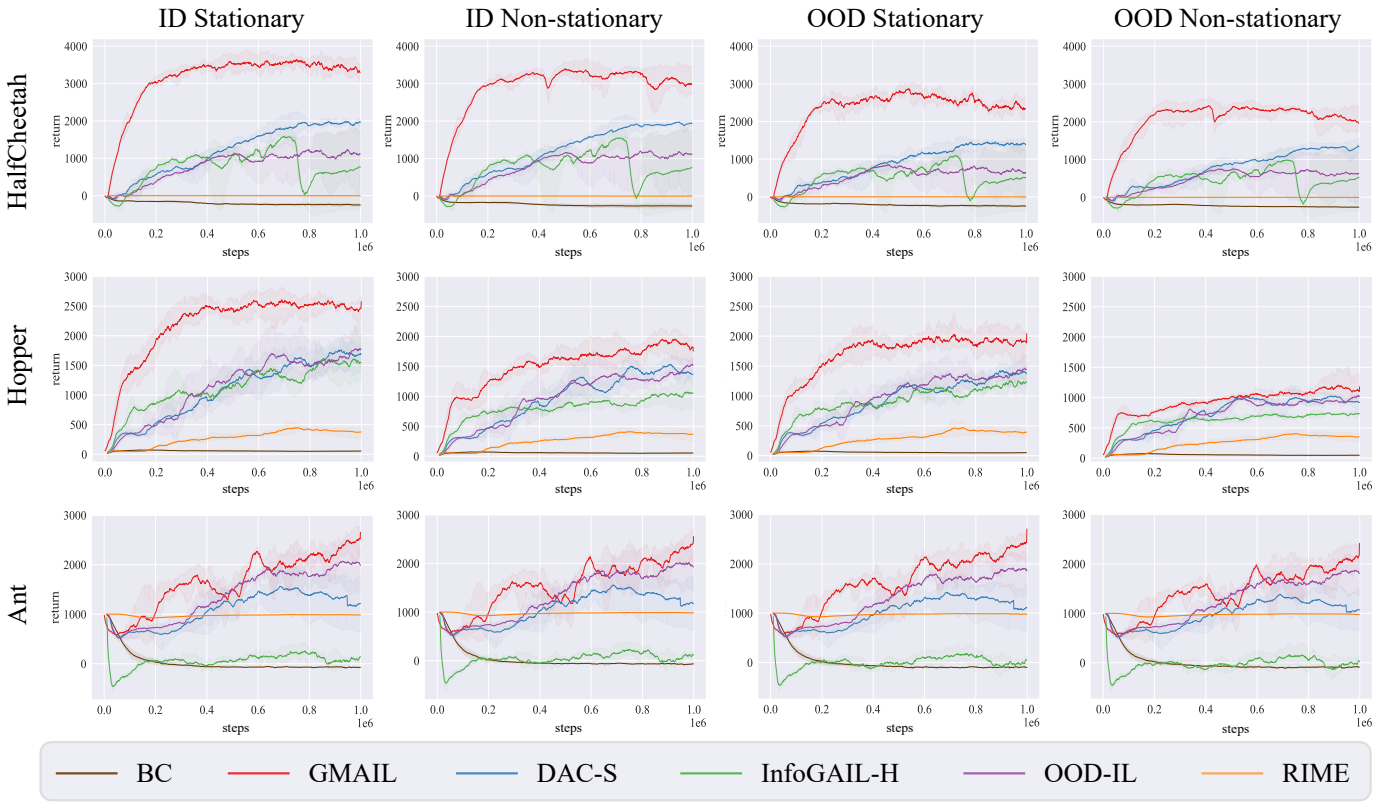

Fig. 8. Learning curves with `body mass` being the varying parameter.

REFERENCES

[1] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[2] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.

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

[4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.

[5] D. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Comput.*, vol. 3, no. 1, pp. 88–97, 1991.

[6] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, "Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning," in *Proc. Int. Conf. Learn. Representations*, 2019.

[7] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1582–1591.

[8] Y. Li, J. Song, and S. Ermon, "InfoGAIL: Interpretable imitation learning from visual demonstrations," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3812–3822.

[9] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. 29th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4565–4573.

[10] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. 29th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2172–2180.

[11] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. MIT Press, 2012.

[12] Y. Qiu, J. Wu, Z. Cao, and M. Long, "Out-of-dynamics imitation learning from multimodal demonstrations," in *Proc. Ann. Conf. Robot. Learn.*, 2022, pp. 1071–1080.

[13] J. Chae, S. Han, W. Jung, M. Cho, S. Choi, and Y. Sung, "Robust imitation learning against variations in environment dynamics," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 2828–2852.

[14] D. M. Endres and J. E. Schindelin, "A new metric for probability distributions," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1858–1860, 2003.