References

[1] Joseph Abbate, R Conlin, and E Kolemen. Data-driven profile prediction for diii-d. *Nuclear Fusion*, 61(4):046027, 2021.

[2] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning*, pages 403–415. PMLR, 2023.

[3] Kei Akuzawa, Yusuke Iwasawa, and Yutaka Matsuo. Estimating disentangled belief about hidden state and hidden task for meta-reinforcement learning. In *Learning for Dynamics and Control*, pages 73–86. PMLR, 2021.

[4] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[5] Lucas Alpi. Pid control: Breaking the time barrier, November 2019. URL `https://www.novusautomation.com/en/article_PID_control#:~:text=In%201911%20the%20first%20PID,still%20widely%20used%20in%20automation.`

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[7] Laszlo Bardoczi, NC Logan, and EJ Strait. Neoclassical tearing mode seeding by nonlinear three-wave interactions in tokamaks. *Physical Review Letters*, 127(5):055002, 2021.

[8] MD Boyer, KG Erickson, BA Grierson, DC Pace, JT Scoville, J Rauch, BJ Crowley, JR Ferron, SR Haskey, DA Humphreys, et al. Feedback control of stored energy and rotation with variable beam energy and perveance on diii-d. *Nuclear Fusion*, 59(7):076004, 2019.

[9] RJ Buttery, RJ La Haye, P Gohil, GL Jackson, H Reimerdes, EJ Strait, and DIII-D Team. The influence of rotation on the β n threshold for the 2/ 1 neoclassical tearing mode in diii-d. *Physics of Plasmas*, 15(5):056115, 2008.

[10] Massimo Caccia, Jonas Mueller, Taesup Kim, Laurent Charlin, and Rasool Fakoor. Task-agnostic continual reinforcement learning: In praise of a simple baseline. *arXiv preprint arXiv:2205.14495*, 2022.

[11] Ian Char, Youngseog Chung, Willie Neiswanger, Kirthevasan Kandasamy, Andrew O Nelson, Mark Boyer, Egemen Kolemen, and Jeff Schneider. Offline contextual bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[12] Ian Char, Joseph Abbate, László Bardóczi, Mark D Boyer, Youngseog Chung, Rory Conlin, Keith Erickson, Viraj Mehta, Nathan Richner, Egemen Kolemen, et al. Offline model-based reinforcement learning for tokamak control. *Machine Learning for the Physical Sciences Workshop NeurIPS 2022*, 2022.

[13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[14] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[15] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897): 414–419, 2022.

[16] Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta learning of exploration. *arXiv preprint arXiv:2008.02598*, 2020.

[17] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[18] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[19] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Robust predictable control. *Advances in Neural Information Processing Systems*, 34:27813–27825, 2021.

[20] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[21] BA Grierson, MA Van Zeeland, JT Scoville, B Crowley, I Bykov, JM Park, WW Heidbrink, A Nagy, SR Haskey, and D Liu. Testing the diii-d co/counter off-axis neutral beam injected power and ability to balance injected torque. *Nuclear Fusion*, 61(11):116049, 2021.

[22] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[23] Sang-hee Hahn, YJ Kim, BG Penaflor, JG Bak, H Han, JS Hong, YM Jeon, JH Jeong, M Joung, JW Juhn, et al. Progress and plan of kstar plasma control system upgrade. *Fusion Engineering and Design*, 112:687–691, 2016.

[24] Dongqi Han, Kenji Doya, and Jun Tani. Variational recurrent models for solving partially observable control tasks. *arXiv preprint arXiv:1912.10703*, 2019.

[25] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

[26] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[28] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pages 2117–2126. PMLR, 2018.

[29] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[31] Yuankun Jiang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. Monotonic robust policy optimization with model discrepancy. In *International Conference on Machine Learning*, pages 4951–4960. PMLR, 2021.

[32] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[33] Andrej Karpathy. nanogpt, 2022–. URL `https://github.com/karpathy/nanoGPT`.

[34] Wenzhe Li, Hao Luo, Zichuan Lin, Chongjie Zhang, Zongqing Lu, and Deheng Ye. A survey on transformers in reinforcement learning. *arXiv preprint arXiv:2301.03044*, 2023.

[35] Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E Gonzalez, Michael I Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. arxiv e-prints, page. *arXiv preprint arXiv:1712.09381*, 2017.

[36] Xingyu Lu, Kimin Lee, Pieter Abbeel, and Stas Tiomkin. Dynamics generalization via information bottleneck in deep reinforcement learning. *arXiv preprint arXiv:2008.00614*, 2020.

[37] La Marzocco. A brief history of the pid, October 2015. URL `https://home.lamarzoccousa.com/history-of-the-pid/`.

[38] Viraj Mehta, Biswajit Paria, Jeff Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning. *arXiv preprint arXiv:2112.05244*, 2021.

[39] Viraj Mehta, Ian Char, Joseph Abbate, Rory Conlin, Mark D Boyer, Stefano Ermon, Jeff Schneider, and Willie Neiswanger. Exploration via planning for information about the optimal trajectory. *arXiv preprint arXiv:2210.04642*, 2022.

[40] Luckeciano C Melo. Transformers are meta-reinforcement learners. In *International Conference on Machine Learning*, pages 15340–15359. PMLR, 2022.

[41] Lingheng Meng, Rob Gorbet, and Dana Kulić. Memory-based deep reinforcement learning for pomdps. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626. IEEE, 2021.

[42] Nicolas Minorsky. Directional stability of automatically steered bodies. *Journal of the American Society for Naval Engineers*, 34(2):280–309, 1922.

[43] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[44] Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free RL can be a strong baseline for many POMDPs. In *International Conference on Machine Learning*, pages 16691–16723. PMLR, 2022.

[45] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–. URL `https://github.com/fmfn/BayesianOptimization`.

[46] Junier B Oliva, Barnabás Póczos, and Jeff Schneider. The statistical recurrent unit. In *International Conference on Machine Learning*, pages 2671–2680. PMLR, 2017.

[47] OpenAI. Gpt-4 technical report. *arXiv*, 2023.

[48] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

[49] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pages 7487–7498. PMLR, 2020.

[50] PA Politzer, CC Petty, RJ Jayakumar, TC Luce, MR Wade, JC DeBoo, JR Ferron, P Gohil, CT Holcomb, AW Hyatt, et al. Influence of toroidal rotation on transport and stability in hybrid scenario plasmas in diii-d. *Nuclear Fusion*, 48(7):075001, 2008.

[51] Vitchyr Pong and Ashvin Nair. rlkit. `https://github.com/rail-berkeley/rlkit`, 2018–.

[52] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[53] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[54] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[55] H Reimerdes, AM Garofalo, GL Jackson, M Okabayashi, EJ Strait, Ming-Sheng Chu, Y In, RJ La Haye, MJ Lanctot, YQ Liu, et al. Reduced critical rotation for resistive-wall mode stabilization in a near-axisymmetric configuration. *Physical review letters*, 98(5):055001, 2007.

[56] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[57] JT Scoville, DA Humphreys, JR Ferron, and P Gohil. Simultaneous feedback control of plasma rotation and stored energy on the diii-d tokamak. *Fusion engineering and design*, 82(5-14): 1045–1050, 2007.

[58] J Seo, Y-S Na, B Kim, CY Lee, MS Park, SJ Park, and YH Lee. Development of an operation trajectory design algorithm for control of multiple 0d parameters using deep reinforcement learning in kstar. *Nuclear Fusion*, 62(8):086049, 2022.

[59] Jaemin Seo, Y-S Na, B Kim, CY Lee, MS Park, SJ Park, and YH Lee. Feedforward beta control in the kstar tokamak by deep reinforcement learning. *Nuclear Fusion*, 61(10):106010, 2021.

[60] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

[61] B Tobias, M Chen, IGJ Classen, CW Domier, R Fitzpatrick, BA Grierson, NC Luhmann Jr, CM Muscatello, M Okabayashi, KEJ Olofsson, et al. Rotation profile flattening and toroidal flow shear reversal due to the coupling of magnetic islands in tokamaks. *Physics of Plasmas*, 23 (5):056107, 2016.

[62] ITER Physics Expert Group on Confinement Transport, , ITER Physics Expert Group on Confinement Modelling Database, , and ITER Physics Basis Editors. Chapter 2: Plasma confinement and transport. *Nuclear Fusion*, 39(12):2175–2249, 1999.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[64] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[65] T Wakatsuki, T Suzuki, N Oyama, and N Hayashi. Ion temperature gradient control using reinforcement learning technique. *Nuclear Fusion*, 61(4):046036, 2021.

[66] ML Walker, DA Humphreys, JA Leuer, JR Ferron, and BG Penaflor. Implementation of model-based multivariable control on diii-d. *GA-A23468*, 2000.

[67] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[68] Zhihan Yang and Hai Huu Nguyen. Recurrent off-policy baselines for memory-based continuous control. In *Deep RL Workshop NeurIPS 2021*, 2021.

[69] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

[70] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

Appendices

A Implementation Details

Code Release All code for implementations are provided in the supplemental material along with instructions for how to run experiments. The only experiment that cannot be run are the "real" cases for tokamak control.

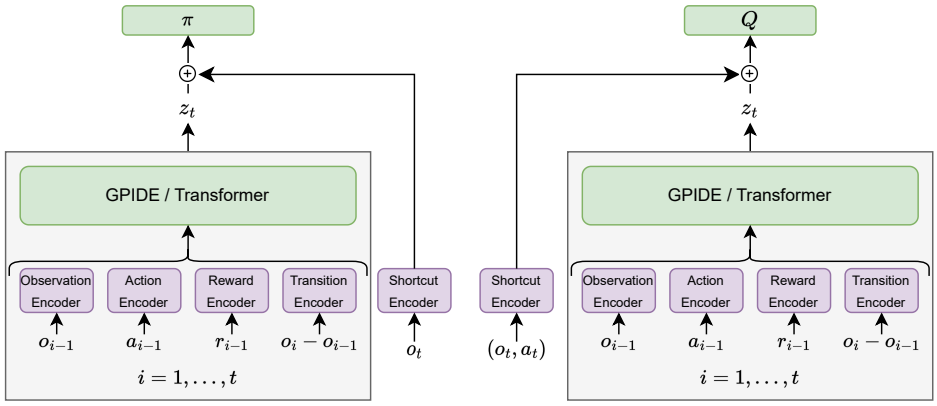

Figure 5: **General Policy and Q Function Architectures.** This architecture is heavily inspired by Ni et al. [44]. The gray box shows the history encoder modules, and this is the only thing that changes between baseline methods in the tracking problems. Note that there are two encoders: one for the policy function and one for the Q value function. The purple boxes show the input encoders, and hyperaparmeters for these can be found in Table 6. We found the shortcut encoders to be essential to good performance. The architecture when using GRU is nearly identical; however, there is no "Transition Encoder" since Ni et al. [44] encodes (o_i, a_{i-1}, r_{i-1}) for each time step instead.

Architecture We use the same general architecture for each of the RL methods in this paper (see Figure 5). Each input to the history encoders, policy functions, and Q-value functions have corresponding encoders. This setup closely follows what was done in Ni et al. [44]. The encoders are simply linear projections; however, in the case of our GRU history encoder we do linear projections followed by a ReLU activation (as done in Ni et al. [44]). Although hypothetically the policy only needs to take in history encoding, z_t, since the current observation, we found it essential for the current observation to be passed in independently and have its own encoder.

A.1 GPIDE Implementation Details

In addition to what is mentioned in Section 3, we found that there were several choices that helped with training. First, there may be some scaling issues because $o_t - o_{t-1}$ may be small or the result of summation type heads may result in large encodings. To account for this, we use batch normalization layers [30] before each input encoding and after each ℓ^h.

There are very few nonlinear components of GPIDE. The only one that remains constant across all experiments is that a tanh activation is used for the final output of the encoder. For tracking tasks, the decoder g_θ has 1 hidden layer with 64 units and uses a ReLU activation function. For PyBullet tasks, g_θ is a linear function.

A.2 Recurrent and Transformer Baseline Details

Recurrent Encoder. For the recurrent encoder, we tried to match as many details as Ni et al. [44] as possible. We double checked our implementation against theirs and confirmed that it achieves similar performance.

Transformer Encoder. We follow the GPT2 architecture [52] for inspiration, and particularly the code provided in Karpathy [33]. In particular, we use a number of multi-headed self-attention blocks in sequence with residual connections. We use layer normalization [6] before multi-headed attention and out projections; however, we do not use dropout. The out projection for each multi-headed self-attention block has one hidden layer with four times the number of units as the embedding dimension. Although Melo [40] suggests using T-Fixup weight initialization, we found that more reliably high performance was achieved with the weight initialization of Radford et al. [52]. Lastly, we used the same representation for the history as GPIDE, i.e. $(o_{t-1}, a_{t-1}, r_{t-1}, o_t - o_{t-1})$, since it results in better performance.

A.3 PID Baseline

To tune our PID baseline, we used Bayesian Optimization over the three (for SISO) or six (for MIMO) dimensional space. Specifically we use the library provided by Nogueira [45]. The output of the blackbox optimization is the average over 100 different settings (independent from the 100 settings used for testing). We allow the optimization procedure to collect as many samples as the RL methods. The final performance reported uses the PID controller with the best gains found during the optimization procedure. The bounds for each of the tracking tasks were eyeballed to be appropriate, which potentially preferably skews performance.

B Hyperaparameters

Because of resource restrictions, we were unable to do full hyperparameter tuning for each benchmark presented in this paper. Instead, we focused on ensuring that all history encoding methods were roughly comparable, e.g. dimension of encoding, number of parameters, etc. Tables 5 show selected hyperparameters, and the following subsections describe how an important subset of these hyperparameters were picked. Any tuning that was done was over three seeds using 100 fixed settings (different from the 100 settings used for testing).

Task Type	Learning Rate	Batch Size	Discount Factor	Policy Network	Q Network	Path Length Encoding
Tracking	$3e^{-4}$	32 (256 for PIDE)	0.95	[24]	[256, 256]	100
PyBullet	$3e^{-4}$	32 (256 for PIDE)	0.99	[256, 256]	[256, 256]	64

Table 5: **SAC Hyperparameters.** The "Path Length Encoding" is the amount of history each encoder gets to observe besides PIDE which, because of the nature of it, uses the entire episode.

	Observation	Action	Reward	Transition	Policy Shortcut	Q Shortcut	History Encoding
GPIDE (Tracking)	8	N/A	N/A	8	8	64	64
GRU (Tracking)	8	N/A	N/A	N/A	8	64	64
Transformer (Tracking)	16	N/A	N/A	16	8	64	64
GPIDE (PyBullet)	32	16	16	64	8	64	128
Transformer (PyBullet)	48	16	16	48	8	64	128

Table 6: **Dimension for the Input Encoders and Final History Encoding**. The input encoders correspond to the output dimensions of the purple boxes in Figure 5. By "History Encoding" size we mean the dimension of z_t.

Task Type	D	g_θ Hidden Size
Tracking	16	[64]
PyBullet	32	[]

Table 7: **GPIDE Specific Hyperparamters.** Recall that D corresponds to the output dimension of f_θ. Empty brackets for the hidden size means that g_θ is a linear function.

B.1 Hyperparamters for Tracking Tasks

For tracking tasks, we tried using a history encoding size of 32 and 64 for GRU, and we found that performance was better with 64. This is surprising since PIDE can perform well in these environments

Task Type	Number of Layers	Number of Heads	Embedding Size per Head
Tracking	2	4	8
PyBullet	4	8	16

Table 8: **Transformer Specific Hyperparamters**

Encoder	SISO Tracking	MIMO Tracking (2D)	PyBullet
Transformer	25,542	25,644	793,868-795,026
GRU	14,240	14,264	74,816-75,248
GPIDE	13,228	13,288	75,296-76,486
GPIDE-ES	12,204	12,264	50,720-51,910
GPIDE-ESS	12,204	12,264	50,720-51,910
GPIDE-Attention	15,276	15,336	99,872-101,062

Table 9: **Number of Parameters in History Encoder Modules**. The number of parameters corresponds to the gray boxes in Figure 5. The difference in SISO vs MIMO and the PyBullet tasks is due to the different observation and action space dimensionalities.

even though its history encoding is much smaller (3 or 6 dimensional). To make it a fair comparison, we set the history encoding dimension for GPIDE and transformer to be 64 as well. We use one layer for GRU. For the transformer-specific hyperparameters we choose half of what appears in the PyBullet tasks.

B.2 Hyperparameters for PyBullet Task

For the PyBullet tasks, we simply tried to emulate most of the hyperparameters found in Ni et al. [44]. For the transformer, we choose to use similar hyperparameters to those found in Melo [40]. However, we found that, unlike the tracking tasks, positional encoding hurts performance. As such, we do not include it for PyBullet experiments.

B.3 Hyperparameters for Ablations

For the ablations of GPIDE, we use $\alpha = 0.01, 0.1, 0.25, 0.5, 0.9, 1.0$ for the smoothing parameters when only exponential smoothing is used. When using exponential smoothing and summation, the $\alpha = 0.01$ head is replaced with a summation head. The attention version of GPIDE replaces all six of these heads with attention.

C Environment Descriptions

C.1 Mass Spring Damper

For both MSD and DMSD, the observations include the current mass position(s), the target reference position(s), and the last action played. Each episode lasts for 100 time steps. For all RL methods, the action is a difference in force applied to the mass, but for the PID the action is simply the force to be applied to the mass at that time. The force is bounded between -10 and 10 N for MSD and -30 and 30 N for DMSD. Each episode, system parameters are drawn from a uniform distribution with bounds shown in Table 10 (they are the same for both MSD and DMSD). Targets are drawn to uniformly at random to be -1.5 to 1.5 m offset from the masses' resting positions.

C.2 Navigation Environment

Like the MSD and DMSD environments, the navigation experiment lasts 100 time steps each episode. Additionally, the observation includes position signal, target locations, and the last action. For all methods we set the action to be the change in force, and the total amount of force is bounded between -10 and 10 N. The penalty on the reward is equal to 0.01 times the magnitude of the change in force. In addition, the maximum magnitude of the velocity for the agent is bounded by $1.0m/s$. The agent

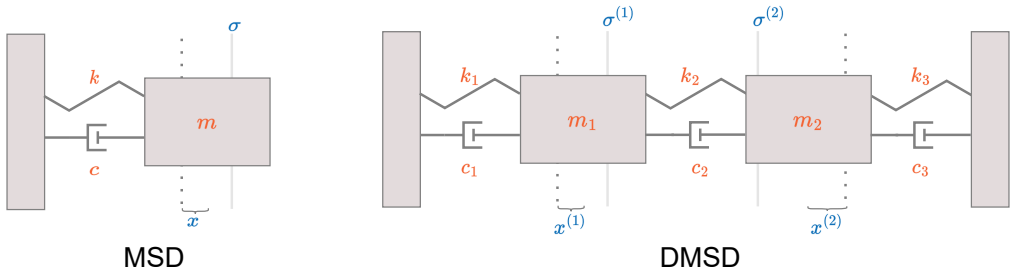

Figure 6: **Diagram of the Mass Spring Damper Environments.** The diagram on the left the Mass Spring Damper (MSD) environment, and the diagram on the right shows the Double Mass Spring Damper (DMSD) environment. In the diagram, we have labelled the system parameters and the parts of the observation. The dotted line shows where the center of the mass is located with no force applied, and the current position of the mass is measured with respect to this point.

System Parameter	Fixed	Small	Large
Damping Constant	$\mathcal{U}(4.0, 4.0)$	$\mathcal{U}(3.5, 5.5)$	$\mathcal{U}(2.0, 10.0)$
Spring Constant	$\mathcal{U}(2.0, 2.0)$	$\mathcal{U}(1.75, 3.0)$	$\mathcal{U}(0.5, 6.0)$
Mass	$\mathcal{U}(20.0, 20.0)$	$\mathcal{U}(17.5, 40.0)$	$\mathcal{U}(10.0, 100.0)$

Table 10: **MSD and DMSD System Parameter Distributions.** Each episode system parameters are uniformly at random drawn from these bounds.

always starts at the location $(0, 0)$, and the target is picked uniformly at random to be within a box of length 10 centered around the origin.

Every episode, the mass, kinetic friction coefficient, and static friction coefficient is sampled, The friction is sampled by first sampling the total amount of friction in the system, and then sampling what proportion of the total friction is static friction. All distributions for the system parameters are uniform, and we show the bounds in Table 11.

C.3 Tokamak Control Environment

Simulator Our simulator version of the tokamak control is inspired by equations used by Boyer et al. [8], Scoville et al. [57]. In particular, we use the following relations for stored energy, E, and rotation, v_{rot}:

$$\dot{E} = P - \frac{E}{\tau_E}$$
$$\dot{v}_{\text{rot}} = C_{\text{rot}} T - \frac{v_{\text{rot}}}{\tau_m}$$

where P is the total power, T is the total torque, τ_E is the energy confinement time, τ_m is the momentum confinement time, and C_{rot} is a quantity relying on the ion density and major radius of the plasma. We treat τ_m and C_{rot} is constants with values of 0.1 and 80.0 respectively.

System Parameter	No Friction	Friction
Total Friction	$\mathcal{U}(0.0, 0.0)$	$\mathcal{U}(0.05, 0.25)$
Static Friction (Proportion)	$\mathcal{U}(0.0, 0.0)$	$\mathcal{U}(0.25, 0.75)$
Mass	$\mathcal{U}(15.0, 25.0)$	$\mathcal{U}(5.0, 35.0)$

Table 11: **Navigation System Parameter Distributions.** Each episode system parameters are uniformly at random drawn from these bounds. The static friction parameter drawn is the proportion of the total friction that is static friction.

Minor Radius (m)	Plasma Current (MA)	Toroidal Magnetic Field (T)
$\mathcal{N}(0.589, 0.02)$	$\mathcal{N}(1e6, 1e5)$	$\mathcal{N}(2.75, 0.1)$

Table 12: **Tokamak Control Simulator Distributions.**

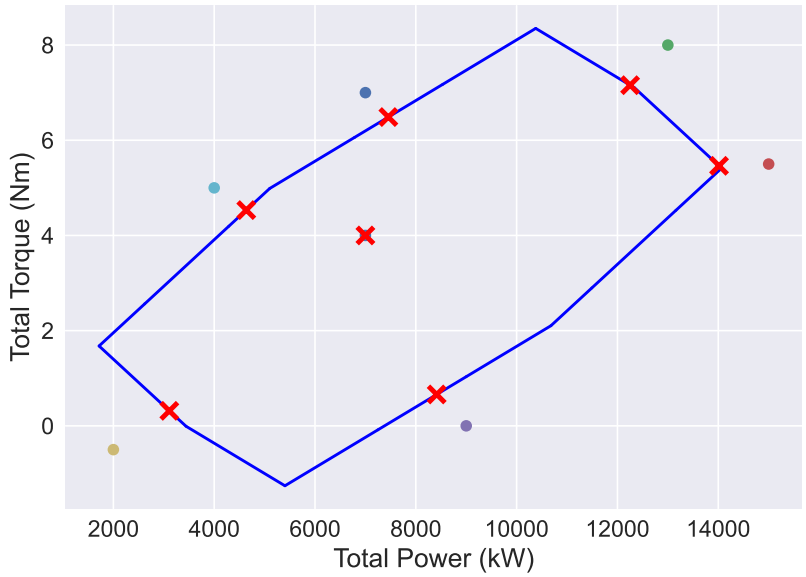

Figure 7: **Power and Torque Bounds.** The region outlined in blue shows the possible power-torque configurations. The dots show possible requests, and the corresponding red X marks show the actual achieved power-torque setting.

We base the energy confinement time off of the ITERH-98 scaling [62]. This uses many measurements of the plasma, but we focus on a subset of these and treat the rest as constants. In particular,

$$\tau_E = C_E I^{0.95} B^{0.15} P^{-0.69}$$

where C_E is a constant value we set to be 200, I is the plasma current, and B is the toroidal magnetic field. To relate the stored energy to β_N we use the rough approximation

$$\beta_N = C_\beta \left(\frac{aB}{I} \right) E$$

where C_β is a constant we set to be 5, and a is the minor radius of the plasma. For a, I, and B, we sample these from the distribution described in Table 12 for each episode. Lastly, we add momentum to the stored energy. That is, the stored energy derivative at time t, $\dot{E}_t$, is

$$\dot{E}_t = 0.5 \left(P_t - \frac{E_t}{\tau_E} \right) + 0.5 \dot{E}_{t-1}$$

The actions for all control methods is the amount of change for the power and torque. Because the total amount of power and torque injected rely on the beams, they are not totally disentangled. In Figure 7, we show the bounds for the action space. Furthermore, we bound the amount that power and torque can be changed by roughly $40MW/s$ and $35Nm/s$, respectively. Each step is 0.025 seconds.

Each episode lasts for 100 increments of 0.025 seconds. The observations are the current β_N and rotation values, their reference values, and the current power and torque settings. We make the initial β_N and rotation relatively small in order to simulate the plasma ramping up. We let the β_N and rotation targets be distributed as $\mathcal{U}(1.75, 2.75)$ and $\mathcal{U}(25.0, 50.0)$ rad/s, respectively.

"Real" For the real versions of the tokamak control experiments, most of the previous (such as action bounds and target distributions) stays the same. The transition function is modeled as a recurrent neural network trained on 7,536 different runs of the DIII-D device. The network uses a GRU, has four hidden layers with 512 units each, and outputs the mean and log variance of a normal distribution describing how β_N and rotation will change. In addition to power and torque, it takes in measurements for the plasma current, the toroidal magnetic field, n1rms (a measurement related to the plasma' stability), and 13 other actuator requests for gas control and plasma shaping. In addition to sampling from the normal distribution outputted by the network, we train an ensemble of ten networks, and an ensemble member is selected every episode. We use five of these models during training and the other five during testing.

Along with an ensemble member being sampled each episode, we also sample a historical run from the dataset, which determines the starting conditions of the plasma and how the other inputs to the neural network which are not modelled evolve over time. Recall that 100 fixed settings are used to evaluate the policy every epoch of training. In this case, a setting consists of targets, an ensemble member, and a historical run.

D Further Results

In this Appendix, we give further evaluation of the evaluation procedure. In addition, we give full tables of results for normalized and unnormalized scores for all methods. We also show performance traces. Note that the percentage changes in Table 4 do not necessarily reflect tables in this section since they report all combinations of environment variants.

D.1 Evaluation Procedure

As stated in the main paper, for tracking tasks, we fix 100 settings (each comprised of targets, start state, and system parameters) that are used to evaluate the policy for every epoch of training (i.e. for every epoch the evaluation returns is the average over all 100 settings returns). We use a separate 100 settings when tuning. For the final returns, we average over the last 10% of recorded evaluations.

For the PyBullet tasks, we use ten different rollouts for evaluation following Ni et al. [44]. We also average over the last 20% of recorded evaluations like they do.

Normalized Table Scores. We now give an in-depth explanation of how the scores in the table are computed. Let $\pi_{(b,i)}$ be the policy trained with baseline method b (e.g. with GPIDE, transformer, or GRU encoder) on environment variant i (e.g. fixed, small, or large). Let $J_j(\pi_{(b,i)})$ be the evaluation of policy $\pi_{(b,i)}$, i.e. the average returns over all seeds and episodes. The normalized score for policy $\pi_{(b,i)}$ on variant j is then

$$\frac{J_j(\pi_{(b,i)}) - \min_{b',i'} J_j(\pi_{(b',i')})}{\max_{b',i'} J_j(\pi_{(b',i')}) - \min_{b',i'} J_j(\pi_{(b',i')})}$$

Note that we only min and max over baseline methods presented in the table.

For PyBullet tasks, we do the same procedure but normalize by the oracle policy's performance (sees both position and velocity) and the Markovian policy's performance (sees only position or velocity but has no history encoder). For both of these policies, we use what was reported from Ni et al. [44]. Note the our normalized scores differ slightly from those used in Ni et al. [44] since they normalize based on the best and worst returns of any policy; however, we believe our scheme gives a more intuitive picture of how any given policy is performing.

D.2 MSD and DMSD Results

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Fixed / Fixed	-6.14 ± 0.02	-5.76 ± 0.02	-5.75 ± 0.01	$\mathbf{-5.69 \pm 0.00}$	-5.76 ± 0.01	-5.75 ± 0.01	-5.73 ± 0.01	-5.83 ± 0.02
Fixed / Small	-7.51 ± 0.04	-7.56 ± 0.03	$\mathbf{-7.29 \pm 0.01}$	-7.37 ± 0.01	-7.33 ± 0.04	-7.37 ± 0.01	-7.32 ± 0.03	-7.39 ± 0.03
Fixed / Large	-11.39 ± 0.09	-12.52 ± 0.11	$\mathbf{-10.87 \pm 0.05}$	-11.44 ± 0.03	-11.61 ± 0.07	-11.48 ± 0.05	-12.50 ± 0.19	-11.52 ± 0.10
Small / Fixed	-6.26 ± 0.06	$\mathbf{-5.80 \pm 0.00}$	-5.92 ± 0.01	-5.95 ± 0.01	-5.93 ± 0.05	-5.89 ± 0.01	-5.92 ± 0.02	-5.91 ± 0.02
Small / Small	-7.49 ± 0.03	$\mathbf{-7.02 \pm 0.01}$	-7.15 ± 0.02	-7.14 ± 0.01	-7.12 ± 0.04	-7.09 ± 0.02	-7.15 ± 0.02	-7.12 ± 0.02
Small / Large	-11.18 ± 0.09	$\mathbf{-9.82 \pm 0.07}$	-10.01 ± 0.03	-10.88 ± 0.04	-10.43 ± 0.14	-10.42 ± 0.13	-10.43 ± 0.12	-10.07 ± 0.14
Large / Fixed	-6.78 ± 0.16	$\mathbf{-6.08 \pm 0.01}$	-6.28 ± 0.03	-6.27 ± 0.01	-6.27 ± 0.03	-6.23 ± 0.04	-6.25 ± 0.04	-6.28 ± 0.05
Large / Small	-7.78 ± 0.12	$\mathbf{-7.25 \pm 0.02}$	-7.44 ± 0.05	-7.43 ± 0.02	-7.45 ± 0.03	-7.44 ± 0.05	-7.44 ± 0.04	-7.48 ± 0.06
Large / Large	-11.12 ± 0.05	$\mathbf{-9.44 \pm 0.02}$	-9.67 ± 0.05	-10.37 ± 0.02	-9.66 ± 0.04	-9.68 ± 0.05	-9.70 ± 0.05	-9.69 ± 0.06
Average	-8.41	-7.92	**-7.82**	-8.06	-7.95	-7.93	-8.05	-7.92

Table 13: **Unnormalized MSD Results**.

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Fixed / Fixed	58.09 ± 1.66	93.18 ± 1.46	94.04 ± 0.83	$\mathbf{100.00 \pm 0.27}$	93.18 ± 1.20	93.77 ± 1.26	96.20 ± 1.22	87.16 ± 1.78
Fixed / Small	36.41 ± 5.36	29.74 ± 3.82	$\mathbf{64.96 \pm 1.38}$	54.90 ± 0.73	59.54 ± 5.78	54.35 ± 1.35	60.89 ± 3.48	51.84 ± 3.38
Fixed / Large	36.58 ± 2.86	0.00 ± 3.42	$\mathbf{53.70 \pm 1.71}$	34.92 ± 0.93	29.55 ± 2.32	33.62 ± 1.71	0.60 ± 6.25	32.51 ± 3.29
Small / Fixed	46.87 ± 5.88	$\mathbf{89.05 \pm 0.32}$	78.21 ± 1.31	75.81 ± 0.79	77.27 ± 4.66	81.41 ± 1.20	78.82 ± 1.44	79.64 ± 1.80
Small / Small	38.25 ± 3.44	$\mathbf{100.00 \pm 0.98}$	83.49 ± 3.07	84.88 ± 0.81	87.78 ± 5.31	90.66 ± 2.02	83.97 ± 2.40	87.57 ± 2.65
Small / Large	43.52 ± 2.82	$\mathbf{87.63 \pm 2.28}$	81.44 ± 0.82	53.21 ± 1.31	68.03 ± 4.43	68.09 ± 4.10	67.78 ± 3.84	79.57 ± 4.71
Large / Fixed	0.00 ± 15.12	$\mathbf{63.36 \pm 1.17}$	45.01 ± 3.18	46.37 ± 1.29	46.68 ± 3.06	49.86 ± 3.84	48.52 ± 3.69	45.03 ± 4.72
Large / Small	0.00 ± 15.75	$\mathbf{70.44 \pm 3.30}$	45.45 ± 6.93	45.73 ± 2.47	43.66 ± 4.47	44.71 ± 6.45	45.21 ± 5.42	39.64 ± 7.82
Large / Large	45.60 ± 1.71	$\mathbf{100.00 \pm 0.61}$	92.60 ± 1.49	69.88 ± 0.69	93.03 ± 1.27	92.36 ± 1.62	91.67 ± 1.68	91.95 ± 1.80
Average	33.92	70.38	**70.99**	62.86	66.53	67.65	63.74	66.10

Table 14: **Normalized MSD Results**.

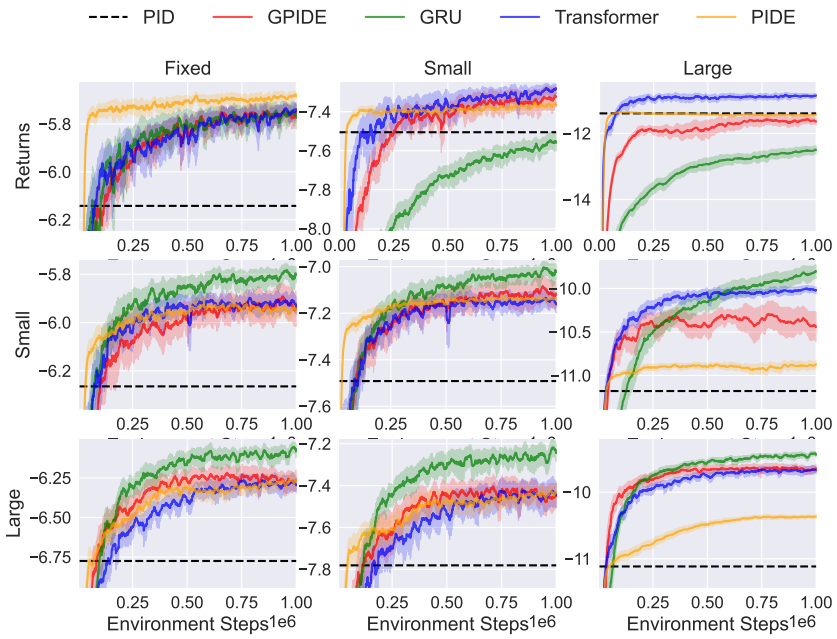

Figure 8: **MSD Performance Curves.** Each row corresponds to a training environment, and each column corresponds to a testing environment.

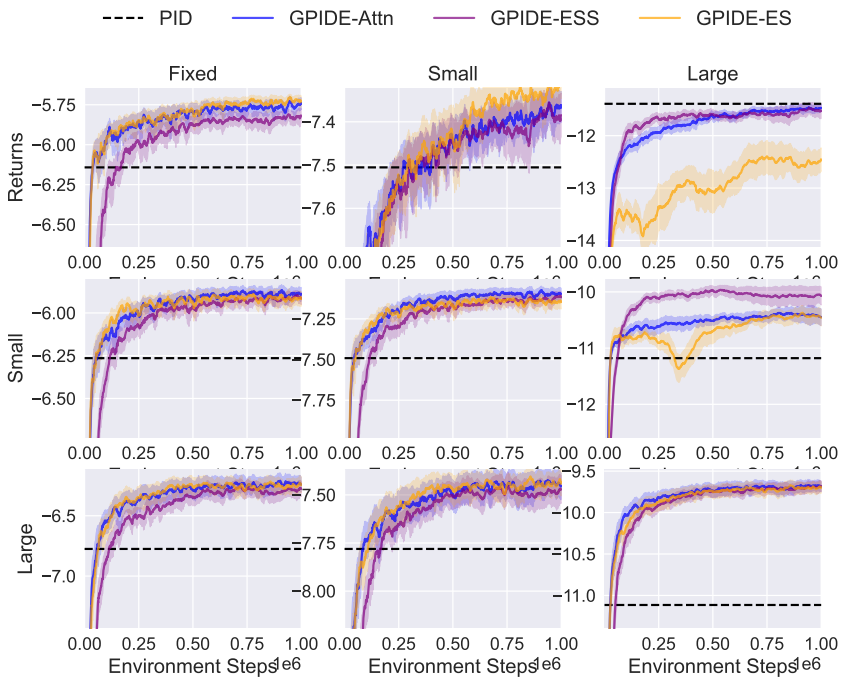

Figure 9: **MSD Performance Curve for Ablations.** Each row corresponds to a training environment, and each column corresponds to a testing environment.

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Fixed / Fixed	-15.33 ± 0.14	-16.20 ± 0.31	-15.41 ± 0.13	$\mathbf{-12.64 \pm 0.04}$	-13.49 ± 0.22	-13.92 ± 0.09	-13.35 ± 0.05	-16.77 ± 0.13
Fixed / Small	-21.29 ± 0.29	-25.21 ± 0.32	-21.37 ± 0.16	$\mathbf{-18.58 \pm 0.05}$	-19.77 ± 0.24	-21.31 ± 0.07	-20.09 ± 0.08	-23.29 ± 0.15
Fixed / Large	-27.59 ± 0.44	-37.21 ± 0.35	-28.16 ± 0.17	$\mathbf{-25.29 \pm 0.18}$	-27.54 ± 0.33	-31.14 ± 0.13	-28.14 ± 0.11	-31.84 ± 0.71
Small / Fixed	-18.15 ± 0.91	-17.75 ± 0.42	-15.86 ± 0.11	$\mathbf{-13.43 \pm 0.09}$	-14.37 ± 0.17	-14.35 ± 0.11	-13.57 ± 0.10	-16.85 ± 0.11
Small / Small	-21.78 ± 0.14	-22.49 ± 0.34	-20.56 ± 0.16	-18.09 ± 0.04	-18.67 ± 0.17	-18.93 ± 0.10	$\mathbf{-17.97 \pm 0.07}$	-21.77 ± 0.10
Small / Large	-26.57 ± 0.22	-31.27 ± 0.36	-26.04 ± 0.24	-23.82 ± 0.13	-23.65 ± 0.20	-23.66 ± 0.10	$\mathbf{-22.72 \pm 0.08}$	-28.26 ± 0.12
Large / Fixed	-21.96 ± 0.62	-22.41 ± 0.32	-18.37 ± 0.30	$\mathbf{-14.83 \pm 0.12}$	-15.75 ± 0.14	-16.79 ± 0.04	-15.23 ± 0.12	-18.89 ± 0.28
Large / Small	-22.30 ± 0.44	-26.63 ± 0.39	-22.00 ± 0.24	$\mathbf{-19.46 \pm 0.08}$	-19.99 ± 0.15	-21.14 ± 0.07	-19.71 ± 0.12	-23.19 ± 0.32
Large / Large	-25.29 ± 0.30	-29.34 ± 0.30	-24.43 ± 0.21	-24.06 ± 0.03	-22.08 ± 0.14	-23.06 ± 0.07	$\mathbf{-21.81 \pm 0.09}$	-25.32 ± 0.19
Average	-22.25	-25.39	-21.36	**-18.91**	-19.48	-20.48	-19.18	-22.91

Table 15: **Unnormalized DMSD Results**.

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Fixed / Fixed	72.45 ± 1.44	63.59 ± 3.16	71.62 ± 1.30	$\mathbf{100.00 \pm 0.39}$	91.35 ± 2.28	86.93 ± 0.89	92.74 ± 0.55	57.75 ± 1.31
Fixed / Small	61.66 ± 3.35	16.43 ± 3.75	60.71 ± 1.80	$\mathbf{93.01 \pm 0.60}$	79.26 ± 2.81	61.50 ± 0.81	75.51 ± 0.97	38.55 ± 1.77
Fixed / Large	62.47 ± 2.86	0.00 ± 2.24	58.78 ± 1.11	$\mathbf{77.38 \pm 1.14}$	62.76 ± 2.13	39.41 ± 0.83	58.92 ± 0.73	34.84 ± 4.61
Small / Fixed	43.59 ± 9.27	47.76 ± 4.25	67.02 ± 1.10	$\mathbf{91.92 \pm 0.90}$	82.32 ± 1.72	82.52 ± 1.14	90.46 ± 0.99	56.98 ± 1.16
Small / Small	56.04 ± 1.57	47.82 ± 3.96	70.07 ± 1.88	98.69 ± 0.48	91.94 ± 2.00	88.95 ± 1.18	$\mathbf{100.00 \pm 0.78}$	56.17 ± 1.11
Small / Large	69.11 ± 1.42	38.57 ± 2.33	72.51 ± 1.58	86.96 ± 0.82	88.08 ± 1.31	87.99 ± 0.64	$\mathbf{94.09 \pm 0.51}$	58.08 ± 0.80
Large / Fixed	4.64 ± 6.34	0.00 ± 3.30	41.37 ± 3.09	$\mathbf{77.62 \pm 1.24}$	68.16 ± 1.45	57.60 ± 0.36	73.51 ± 1.24	36.06 ± 2.85
Large / Small	50.02 ± 5.07	0.00 ± 4.56	53.45 ± 2.80	$\mathbf{82.77 \pm 0.98}$	76.66 ± 1.75	63.36 ± 0.85	79.93 ± 1.43	39.74 ± 3.65
Large / Large	77.38 ± 1.93	51.09 ± 1.98	82.96 ± 1.38	85.37 ± 0.18	98.23 ± 0.90	91.86 ± 0.44	$\mathbf{100.00 \pm 0.56}$	77.22 ± 1.21
Average	55.26	29.47	64.28	**88.19**	82.08	73.35	85.02	50.60

Table 16: **Normalized DMSD Results**.

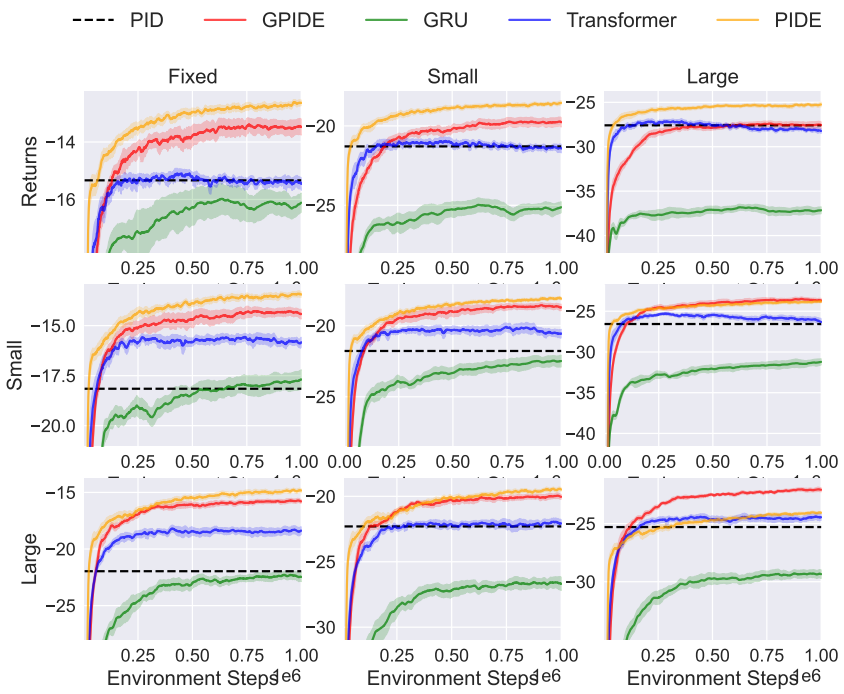

Figure 10: **DMSD Performance Curves.** Each row corresponds to a training environment, and each column corresponds to a testing environment.

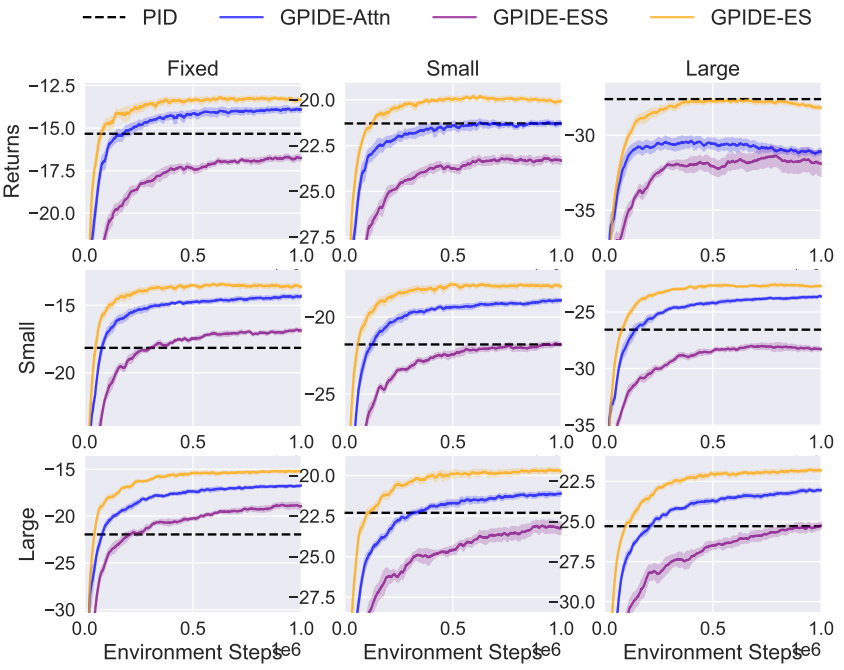

Figure 11: **DMSD Performance Curve for Ablations.** Each row corresponds to a training environment, and each column corresponds to a testing environment.

D.3 Navigation Results

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Sim / Sim	28.94 ± 3.63	96.76 ± 0.15	99.57 ± 0.12	98.33 ± 0.06	$\mathbf{100.00 \pm 0.07}$	99.64 ± 0.06	99.66 ± 0.06	99.81 ± 0.09
Sim / Real	43.12 ± 2.08	0.00 ± 3.94	50.55 ± 0.78	$\mathbf{68.34 \pm 0.57}$	62.16 ± 0.89	63.17 ± 0.57	59.21 ± 1.15	52.52 ± 0.50
Real / Sim	0.00 ± 4.09	57.49 ± 1.17	68.03 ± 0.40	59.54 ± 0.85	$\mathbf{74.88 \pm 0.61}$	72.84 ± 0.64	74.75 ± 0.68	71.13 ± 0.72
Real / Real	67.28 ± 2.05	97.29 ± 0.20	99.20 ± 0.14	95.94 ± 0.04	$\mathbf{100.00 \pm 0.21}$	99.19 ± 0.09	99.11 ± 0.21	99.67 ± 0.17
Average	34.83	62.89	79.34	80.54	**84.26**	83.71	83.18	80.78

Table 17: **Normalized Navigation Results**. Note that these results are after 1 million collected samples.

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Sim / Sim	-17.23 ± 0.18	-13.82 ± 0.01	-13.68 ± 0.01	-13.74 ± 0.00	$\mathbf{-13.65 \pm 0.00}$	-13.67 ± 0.00	-13.67 ± 0.00	-13.66 ± 0.00
Sim / Real	-23.87 ± 0.29	-29.85 ± 0.55	-22.84 ± 0.11	$\mathbf{-20.37 \pm 0.08}$	-21.23 ± 0.12	-21.09 ± 0.08	-21.64 ± 0.16	-22.57 ± 0.07
Real / Sim	-18.69 ± 0.21	-15.79 ± 0.06	-15.26 ± 0.02	-15.69 ± 0.04	$\mathbf{-14.92 \pm 0.03}$	-15.02 ± 0.03	-14.93 ± 0.03	-15.11 ± 0.04
Real / Real	-20.52 ± 0.28	-16.36 ± 0.03	-16.09 ± 0.02	-16.55 ± 0.01	$\mathbf{-15.98 \pm 0.03}$	-16.09 ± 0.01	-16.11 ± 0.03	-16.03 ± 0.02
Average	-20.08	-18.96	-16.97	-16.59	**-16.45**	-16.47	-16.59	-16.84

Table 18: **Unnormalized Navigation Results**. Note that these results are after 1 million collected samples.

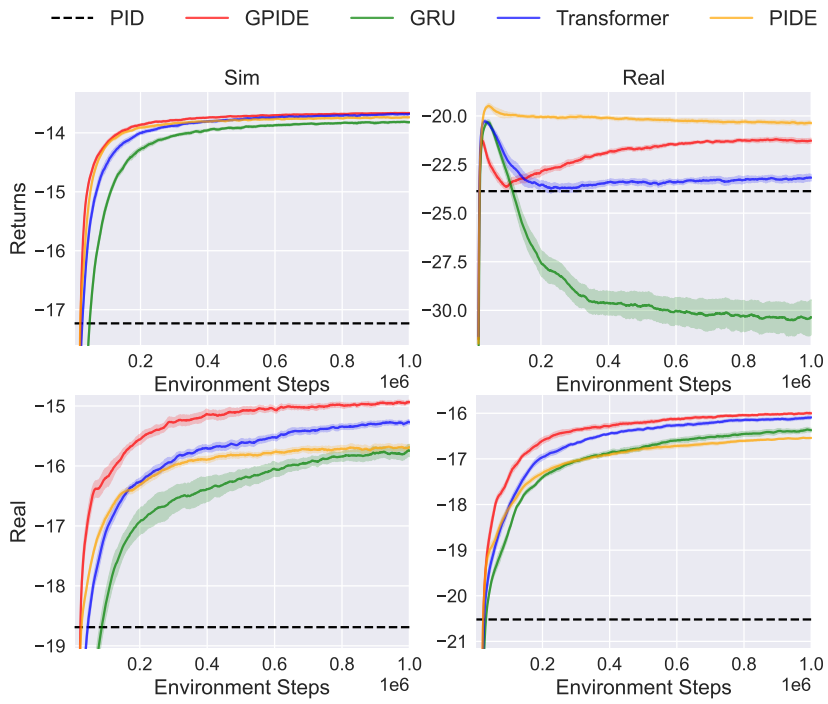

Figure 12: **Navigation Performance Curves.** Each row corresponds to a training environment, and each column corresponds to a testing environment. Note that these runs are only done for one million transitions.

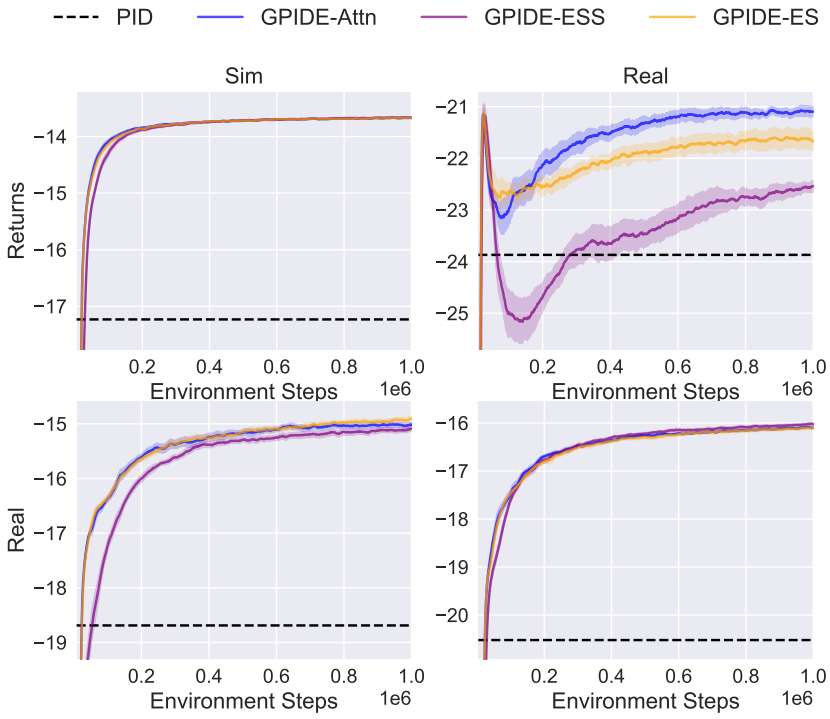

Figure 13: **Navigation Performance Curve for Ablations.** Each row corresponds to a training environment, and each column corresponds to a testing environment. Note that these runs are only done for one million transitions.

D.4 Tokamak Control Results

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Sim / Sim	90.95 ± 0.05	$\mathbf{100.00 \pm 0.03}$	99.63 ± 0.03	84.74 ± 0.16	99.75 ± 0.06	99.91 ± 0.02	99.90 ± 0.02	99.47 ± 0.04
Sim / Real	$\mathbf{89.15 \pm 0.99}$	40.96 ± 5.45	40.05 ± 11.91	0.00 ± 21.04	55.21 ± 4.44	61.56 ± 7.40	65.65 ± 5.66	35.66 ± 4.41
Real / Sim	50.62 ± 3.96	36.33 ± 3.61	35.26 ± 2.22	0.00 ± 3.48	48.40 ± 4.04	52.62 ± 1.38	$\mathbf{56.30 \pm 2.25}$	16.33 ± 5.98
Real / Real	98.45 ± 0.77	98.24 ± 0.38	98.74 ± 0.29	$\mathbf{100.00 \pm 0.23}$	99.30 ± 0.64	98.39 ± 0.33	98.55 ± 0.33	98.27 ± 0.37
Average	**82.29**	68.88	68.42	46.18	75.67	78.12	80.10	62.43

Table 19: **Normalized β_N Tracking Results.**

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Sim / Sim	-8.09 ± 0.00	$\mathbf{-7.19 \pm 0.00}$	-7.22 ± 0.00	-8.71 ± 0.02	-7.21 ± 0.01	-7.19 ± 0.00	-7.20 ± 0.00	-7.24 ± 0.00
Sim / Real	$\mathbf{-16.41 \pm 0.30}$	-31.21 ± 1.67	-31.49 ± 3.66	-43.78 ± 6.46	-26.83 ± 1.36	-24.88 ± 2.27	-23.63 ± 1.74	-32.83 ± 1.35
Real / Sim	-12.12 ± 0.40	-13.55 ± 0.36	-13.66 ± 0.22	-17.18 ± 0.35	-12.34 ± 0.40	-11.92 ± 0.14	$\mathbf{-11.55 \pm 0.22}$	-15.55 ± 0.60
Real / Real	-13.56 ± 0.23	-13.62 ± 0.12	-13.47 ± 0.09	$\mathbf{-13.08 \pm 0.07}$	-13.30 ± 0.20	-13.58 ± 0.10	-13.53 ± 0.10	-13.61 ± 0.11
Average	**-12.55**	-16.39	-16.46	-20.69	-14.92	-14.39	-13.98	-17.31

Table 20: **Unnormalized β_N Tracking Results.**

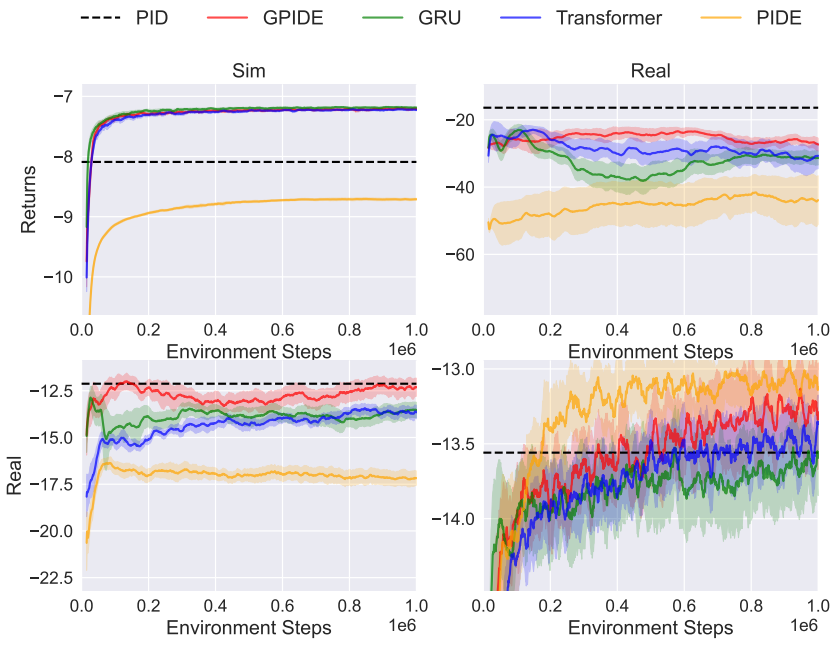

Figure 14: **β_N Tracking Performance Curves.** Each row corresponds to a training environment, and each column corresponds to a testing environment.

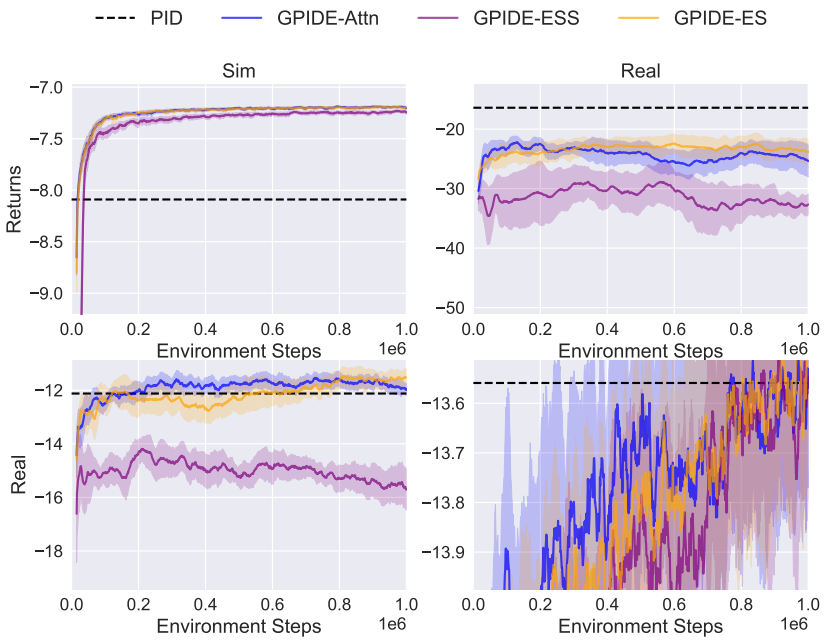

Figure 15: β_N **Tracking Performance Curve for Ablations.** Each row corresponds to a training environment, and each column corresponds to a testing environment.

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Sim / Sim	46.78 ± 0.44	99.50 ± 0.12	97.99 ± 0.50	82.96 ± 0.27	$\mathbf{100.00 \pm 0.15}$	99.64 ± 0.19	99.97 ± 0.12	96.18 ± 1.35
Sim / Real	$\mathbf{83.48 \pm 2.63}$	39.65 ± 5.83	33.22 ± 0.69	0.00 ± 8.87	50.86 ± 1.92	54.36 ± 2.07	52.56 ± 2.44	42.51 ± 2.97
Real / Sim	0.00 ± 8.79	21.31 ± 2.45	7.23 ± 3.86	22.49 ± 1.84	19.02 ± 3.88	$\mathbf{22.70 \pm 4.42}$	5.20 ± 20.06	15.35 ± 8.29
Real / Real	91.76 ± 0.84	98.07 ± 0.52	96.05 ± 0.31	97.94 ± 0.23	99.73 ± 0.46	97.62 ± 0.46	$\mathbf{100.00 \pm 0.28}$	96.33 ± 0.47
Average	55.51	64.63	58.62	50.85	67.40	**68.58**	64.43	62.59

Table 21: **Normalized β_N-Rotation Tracking Results**.

	PID Controller	GRU	Transformer	PIDE	GPIDE	GPIDE-ES	GPIDE-ESS	GPIDE-Attn
Sim / Sim	-27.56 ± 0.08	-18.53 ± 0.02	-18.79 ± 0.09	-21.36 ± 0.05	$\mathbf{-18.45 \pm 0.03}$	-18.51 ± 0.03	-18.45 ± 0.02	-19.10 ± 0.23
Sim / Real	$\mathbf{-30.08 \pm 0.95}$	-45.91 ± 2.10	-48.23 ± 0.25	-60.23 ± 3.20	-41.86 ± 0.69	-40.60 ± 0.75	-41.25 ± 0.88	-44.88 ± 1.07
Real / Sim	-35.57 ± 1.50	-31.92 ± 0.42	-34.33 ± 0.66	-31.72 ± 0.32	-32.31 ± 0.66	$\mathbf{-31.68 \pm 0.76}$	-34.68 ± 3.43	-32.94 ± 1.42
Real / Real	-27.09 ± 0.30	-24.81 ± 0.19	-25.54 ± 0.11	-24.86 ± 0.08	-24.21 ± 0.16	-24.98 ± 0.17	$\mathbf{-24.12 \pm 0.10}$	-25.44 ± 0.17
Average	-30.08	-30.29	-31.72	-34.54	-29.21	**-28.94**	-29.62	-30.59

Table 22: **Unnormalized β_N-Rotation Tracking Results**.

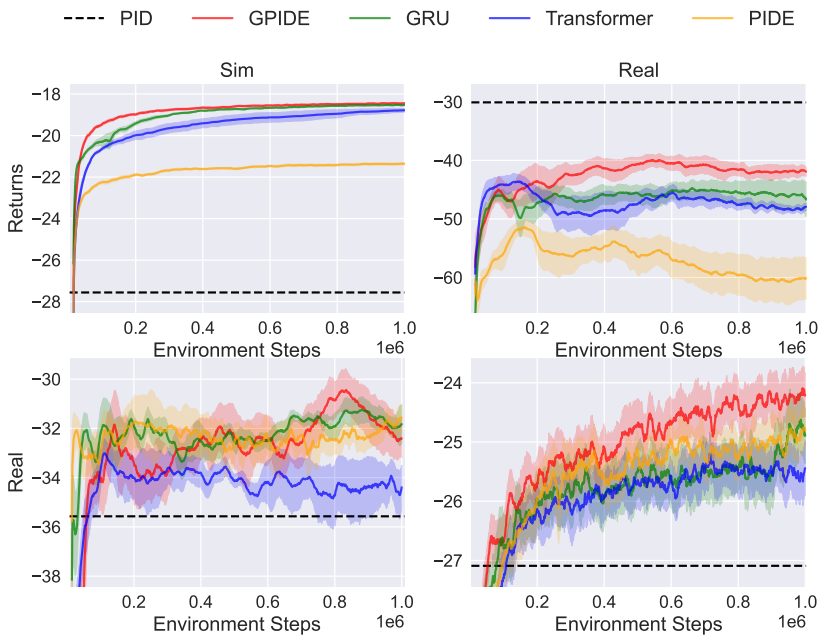

Figure 16: β_N**-Rotation Tracking Performance Curves.** Each row corresponds to a training environment, and each column corresponds to a testing environment.

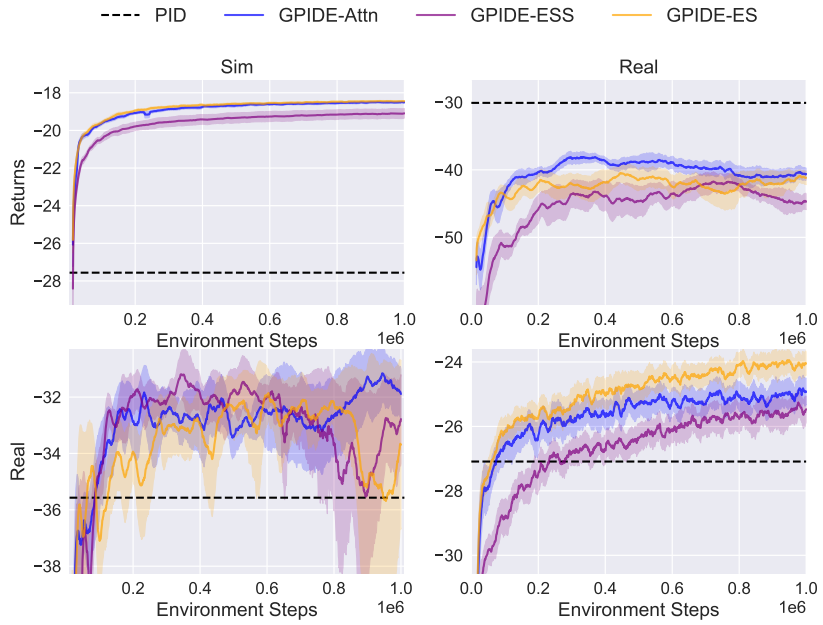

Figure 17: β_N**-Rotation Tracking Performance Curve for Ablations.** Each row corresponds to a training environment, and each column corresponds to a testing environment.

D.5 PyBullet Results

For these results, SAC encodes observations, actions and rewards. TD3 encodes observations and actions since it is the best performing on average.

	PPO-GRU	A2C-GRU	SAC-LSTM	TD3-GRU	VRM	SAC-Transformer	SAC-GPIDE	SAC-GPIDE-ES	SAC-GPIDE-ESS	SAC-GPIDE-Attn
HalfCheetah-P	27.09 ± 7.85	-22.00 ± 5.13	77.06 ± 7.96	$\mathbf{85.80 \pm 5.15}$	-107.00 ± 1.39	37.00 ± 9.97	82.63 ± 3.46	85.45 ± 4.83	71.08 ± 6.95	77.63 ± 4.60
Hopper-P	49.00 ± 5.22	-2.29 ± 3.33	64.36 ± 9.94	84.63 ± 8.33	3.53 ± 1.63	59.54 ± 19.64	93.27 ± 13.56	111.48 ± 4.20	$\mathbf{113.95 \pm 4.67}$	104.87 ± 8.76
Walker-P	1.67 ± 4.39	-10.48 ± 1.73	40.92 ± 15.56	29.08 ± 9.67	-3.89 ± 1.25	24.89 ± 14.80	$\mathbf{96.61 \pm 1.60}$	76.58 ± 5.47	94.58 ± 11.00	71.36 ± 6.57
Ant-P	39.48 ± 3.74	-13.06 ± 6.52	60.97 ± 3.54	-36.36 ± 3.35	-36.39 ± 0.17	-10.57 ± 2.34	$\mathbf{66.66 \pm 2.94}$	64.73 ± 3.82	57.78 ± 3.78	63.19 ± 5.32
HalfCheetah-V	19.68 ± 11.71	-50.13 ± 9.50	18.54 ± 33.09	$\mathbf{59.03 \pm 2.88}$	-80.49 ± 2.97	-41.31 ± 26.15	20.39 ± 29.60	51.03 ± 13.93	53.14 ± 5.86	-54.70 ± 19.89
Hopper-V	13.86 ± 4.80	-0.60 ± 3.33	16.26 ± 12.44	57.43 ± 8.63	10.08 ± 3.51	0.28 ± 8.49	$\mathbf{90.98 \pm 4.28}$	72.63 ± 19.28	90.09 ± 2.50	30.73 ± 1.60
Walker-V	8.12 ± 5.43	-8.02 ± 0.57	-1.57 ± 1.88	-4.63 ± 1.30	-1.80 ± 0.70	-8.21 ± 1.31	36.90 ± 16.59	$\mathbf{68.30 \pm 4.33}$	67.54 ± 3.60	14.85 ± 11.26
Ant-V	1.43 ± 3.26	-13.67 ± 1.83	-16.95 ± 1.29	17.03 ± 6.55	-13.41 ± 0.12	0.81 ± 1.31	$\mathbf{18.03 \pm 5.10}$	4.56 ± 5.20	12.85 ± 1.67	-1.84 ± 5.76
Average	20.04	-15.03	32.45	36.50	-28.67	7.80	63.18	66.84	**70.13**	38.26

Table 23: **Normalized PyBullet Scores**.

	PPO-GRU	A2C-GRU	SAC-LSTM	TD3-GRU	VRM	SAC-Transformer	SAC-GPIDE	SAC-GPIDE-ES	SAC-GPIDE-ESS	SAC-GPIDE-Attn
HalfCheetah-P	1445.81 ± 166.79	403.35 ± 108.97	2506.88 ± 168.93	$\mathbf{2692.53 \pm 109.43}$	-1401.67 ± 29.62	1656.13 ± 211.75	2625.13 ± 73.49	2684.98 ± 102.57	2379.79 ± 147.67	2519.06 ± 97.72
Hopper-P	1436.43 ± 102.09	433.19 ± 65.09	1736.81 ± 194.51	2133.42 ± 162.93	546.93 ± 31.81	1642.63 ± 384.10	2302.31 ± 265.21	2658.48 ± 82.18	$\mathbf{2706.81 \pm 91.39}$	2529.31 ± 171.41
Walker-P	501.06 ± 76.99	288.10 ± 30.39	1189.28 ± 272.77	981.63 ± 169.46	403.60 ± 21.85	908.17 ± 259.52	$\mathbf{2165.52 \pm 28.10}$	1814.40 ± 95.91	2129.91 ± 192.87	1722.81 ± 115.22
Ant-P	2025.52 ± 84.58	837.57 ± 147.53	2511.54 ± 80.13	310.72 ± 75.68	310.24 ± 3.83	893.84 ± 52.83	$\mathbf{2640.16 \pm 66.46}$	2596.63 ± 86.26	2439.48 ± 85.37	2561.67 ± 120.21
HalfCheetah-V	1005.13 ± 289.84	-723.40 ± 235.29	977.02 ± 819.24	$\mathbf{1979.56 \pm 71.40}$	-1475.15 ± 73.42	-505.00 ± 647.43	1022.93 ± 732.93	1781.36 ± 344.95	1833.60 ± 145.14	-836.47 ± 492.45
Hopper-V	534.05 ± 105.85	215.22 ± 73.48	587.10 ± 274.42	1495.11 ± 190.42	450.77 ± 77.35	234.49 ± 187.36	$\mathbf{2235.02 \pm 94.45}$	1830.26 ± 425.16	2215.47 ± 55.16	906.05 ± 35.29
Walker-V	377.80 ± 109.11	53.25 ± 11.45	182.97 ± 37.89	121.44 ± 26.14	178.28 ± 14.09	49.32 ± 26.43	956.43 ± 333.46	$\mathbf{1587.56 \pm 87.15}$	1572.41 ± 72.46	513.07 ± 226.34
Ant-V	684.36 ± 89.48	269.32 ± 50.35	178.98 ± 35.57	1113.19 ± 179.93	276.33 ± 3.18	667.20 ± 35.98	$\mathbf{1140.73 \pm 140.22}$	770.51 ± 143.02	998.35 ± 46.04	594.54 ± 158.48
Average	1001.27	222.08	1233.82	1353.45	-88.84	693.35	1886.03	1965.52	**2034.48**	1313.76

Table 24: **Unnormalized PyBullet Scores**.

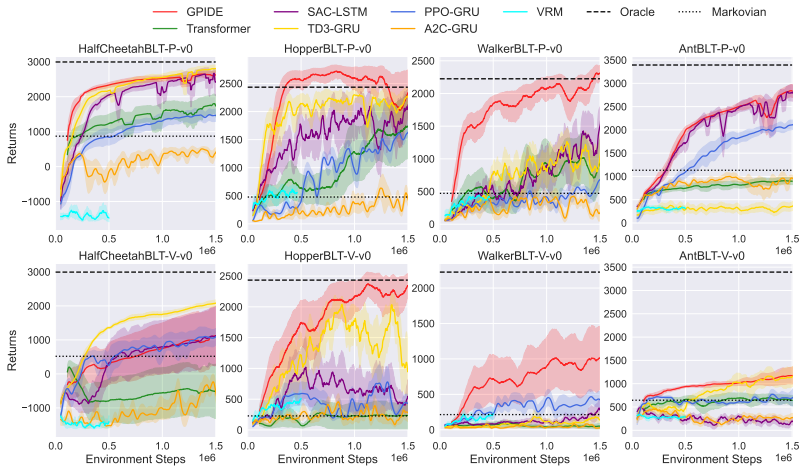

Figure 18: **PyBullet Performance Curves.**

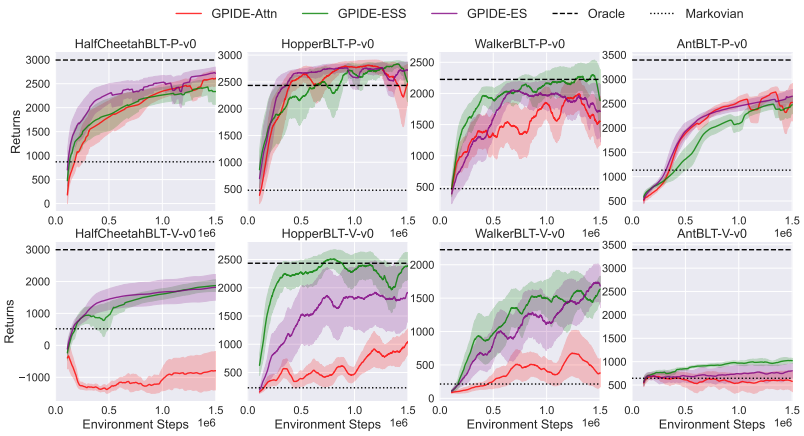

Figure 19: **PyBullet Performance Curve for Ablations.**

Interestingly, we found that GPIDE policies often outperform the oracle policy on Hopper-P. While the oracle performance here was taken from Ni et al. [44], we confirmed this also happens with our own implementation of an oracle policy. We hypothesize that this may be due to the fact the GPIDE policy gets to see actions and rewards and the oracle does not.

D.6 Attention Scheme Visualizations

We generate the attention visualizations (as seen in Figure 4) by doing a handful of rollouts with a GPIDE policy using only attention heads. During this rollout we collect all of the weighting schemes, i.e. softmax $\left(\frac{q_{1:t}k_{1:t}^T}{\sqrt{D}}\right)$, generated throughout the rollouts and average them together. Below, we show additional attention visualizations. In all figures, each plot shows one of the different six heads. For each of these, the policies were evaluated on the same version of the environment they were trained on.

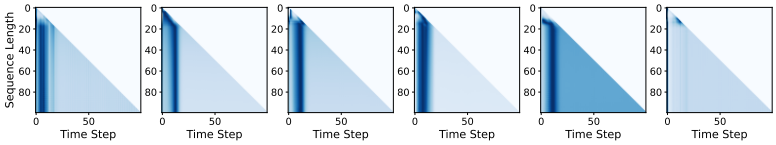

Figure 20: **MSD-Fixed Attention**.

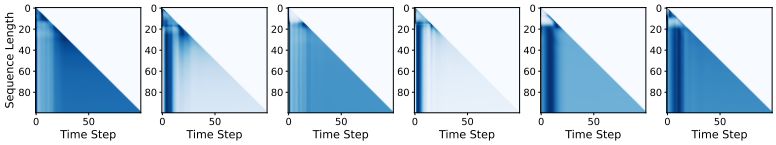

Figure 21: **MSD-Small Attention**.

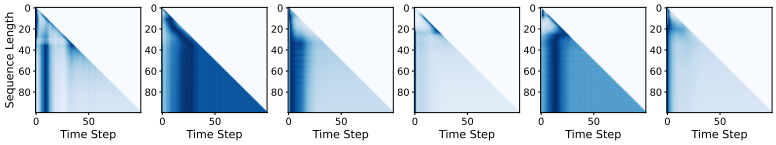

Figure 22: **MSD-Large Attention**.

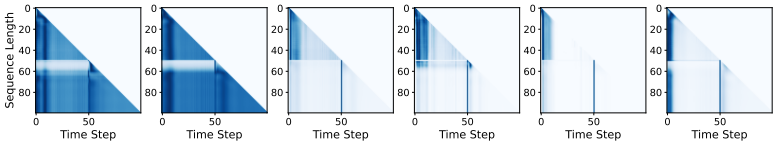

Figure 23: **DMSD-Fixed Attention**.

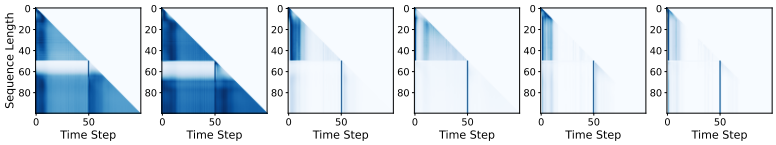

Figure 24: **DMSD-Small Attention**.

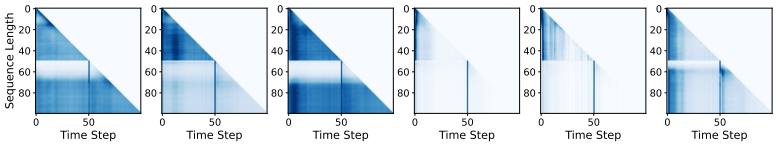

Figure 25: **DMSD-Large Attention**.

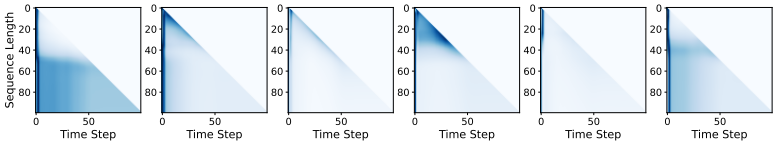

Figure 26: **Navigation No Friction Attention**.

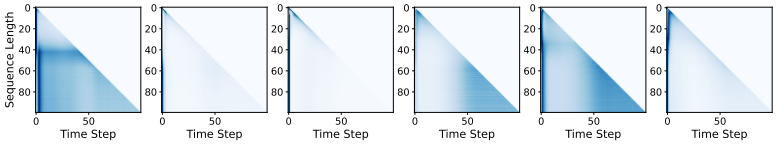

Figure 27: **Navigation Friction Attention**.

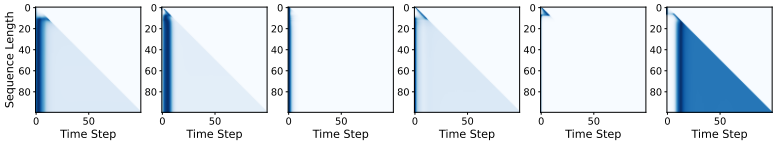

Figure 28: β_N **Tracking Sim Attention**.

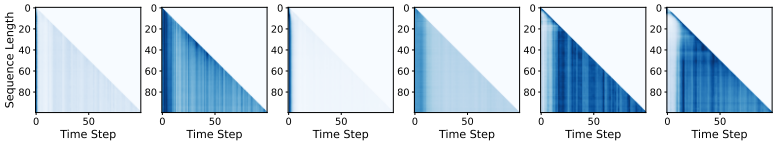

Figure 29: β_N **Tracking Rotation Attention**.

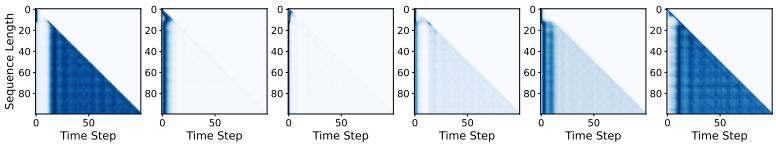

Figure 30: β_N**-Rotation Tracking Sim Attention**.

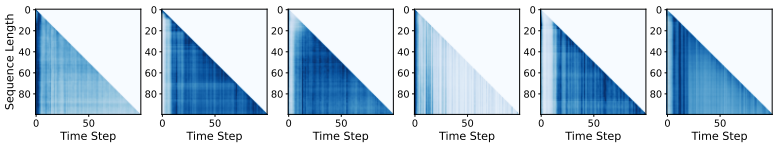

Figure 31: β_N**-Rotation Tracking Rotation Attention**.

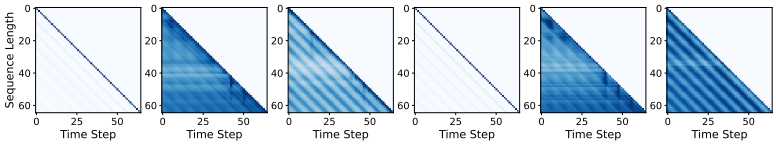

Figure 32: **HalfCheetah-P Attention**.

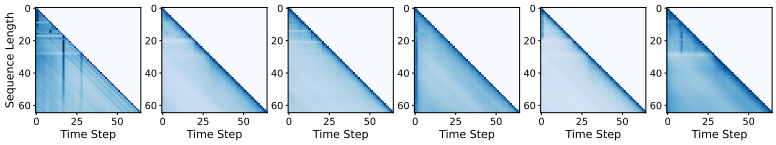

Figure 33: **HalfCheetah-V Attention**.

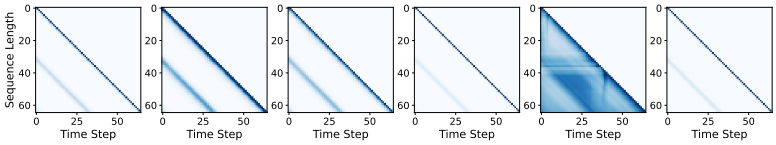

Figure 34: **Hopper-P Attention**.

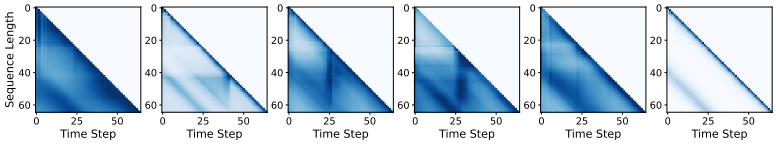

Figure 35: **Hopper-V Attention**.

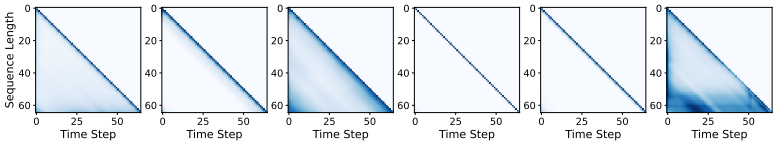

Figure 36: **Walker-P Attention**.

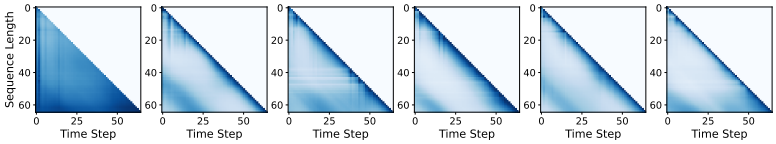

Figure 37: **Walker-V Attention**.

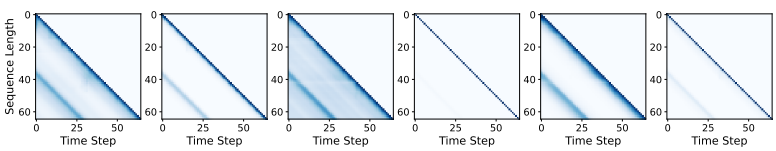

Figure 38: **Ant-P Attention**.

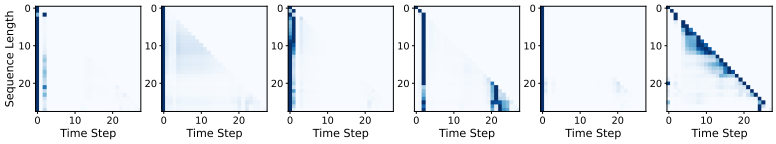

Figure 39: **Ant-V Attention**. Note that total path length is less than 64 here since the agent falls down pretty fast.

D.7 Experiments Using VIB + GRU

As shown in this work, there are often where using a GRU especially results in a policy that is not robust to changes in the dynamics. One may wonder whether using other robust RL techniques is able to mask this inadequacy of GRU. To test this, we look at adding Variational Information Bottlenecking (VIB) to our GRU baseline [4]. Previous works applying this concept to RL usually do not consider the same class of POMDPs as us [36, 29]; however, Eysenbach et al. [19] does have a baseline that uses VIB with a recurrent policy.

To use VIB with RL, we alter the policy network so that it encodes input to a latent random variable, and the decodes into an action. Following the notation of Lu et al. [36], let this latent random variable be Z and the random variable representing the input of the network be S. The goal is to learn a policy that maximizes $J(\pi)$ subject to $I(Z, S) \leq I_C$, where $I(Z, S)$ is the mutual information between Z and S, and I_C is some given threshold. In practice, we optimize the Lagrangian. Where β is a Lagrangian multiplier, $p(Z|S)$ is the conditional density of Z outputted by the encoder, and $q(Z)$ is the prior, the penalizer is $-\beta \mathbb{E}_S \left[D_{\text{KL}} \left(p(Z|S) || q(Z) \right) \right]$. Like other works, we assume that $q(Z)$ is a standard multivarite normal.

We alter our GRU baseline for tracking tasks so that the policy uses VIB. This is not entirely straightforward since our policy network is already quite small. We choose to keep as close to original policy architecture as possible and set the dimension of the latent variable, Z, to be 24. Note that this change has no affect on the history encoder; this only affects the policy network. For our experiments, we set $\beta = 0.1$, but we note that we may be able to achieve better performance through more careful tuning or annealing of β.

In any case, we do see that VIB helps with robustness in many instances (see Table 25). However, the cases where there are improvements are instances where the GRU policy already did a good job at generalizing to the test environment. These are primarily the MSD and DMSD environments where the system parameters drawn during training time are simply a subset of those drawn during testing time. Interestingly, this notion of dynamics generalization matches the set up of the experiments presented in Lu et al. [36]. Surprisingly, in the navigation and tokamak control experiments, where there are more complex differences between the train and test environments, VIB can sometimes hurt the final performance.

	PID Controller	GRU	GRU+VIB	Transformer	PIDE	GPIDE
MSD Fixed / Fixed	-6.14 ± 0.02	-5.76 ± 0.02	-5.73 ± 0.01	-5.75 ± 0.01	$\mathbf{-5.69 \pm 0.00}$	-5.76 ± 0.01
MSD Fixed / Large	-11.39 ± 0.09	-12.52 ± 0.11	-12.50 ± 0.14	$\mathbf{-10.87 \pm 0.05}$	-11.44 ± 0.03	-11.61 ± 0.07
MSD Small / Small	-7.49 ± 0.03	-7.02 ± 0.01	$\mathbf{-7.01 \pm 0.01}$	-7.15 ± 0.02	-7.14 ± 0.01	-7.12 ± 0.04
MSD Small / Large	-11.18 ± 0.09	-9.82 ± 0.07	$\mathbf{-9.57 \pm 0.03}$	-10.01 ± 0.03	-10.88 ± 0.04	-10.43 ± 0.14
DMSD Fixed / Fixed	-15.33 ± 0.14	-16.20 ± 0.31	-15.83 ± 0.28	-15.41 ± 0.13	$\mathbf{-12.64 \pm 0.04}$	-13.49 ± 0.22
DMSD Fixed / Large	-27.59 ± 0.44	-37.21 ± 0.35	-35.34 ± 0.28	-28.16 ± 0.17	$\mathbf{-25.29 \pm 0.18}$	-27.54 ± 0.33
DMSD Small / Small	-21.78 ± 0.14	-22.49 ± 0.34	-22.51 ± 0.24	-20.56 ± 0.16	$\mathbf{-18.09 \pm 0.04}$	-18.67 ± 0.17
DMSD Small / Large	-26.57 ± 0.22	-31.27 ± 0.36	-30.93 ± 0.34	-26.04 ± 0.24	-23.82 ± 0.13	$\mathbf{-23.65 \pm 0.20}$
Nav Sim / Sim	-17.23 ± 0.18	-13.82 ± 0.01	-14.69 ± 0.02	-13.68 ± 0.01	-13.74 ± 0.00	$\mathbf{-13.65 \pm 0.00}$
Nav Sim / Real	-23.87 ± 0.29	-29.85 ± 0.55	-39.57 ± 0.24	-22.84 ± 0.11	$\mathbf{-20.37 \pm 0.08}$	-21.23 ± 0.12
β_N Sim / Sim	-8.09 ± 0.00	$\mathbf{-7.19 \pm 0.00}$	-7.24 ± 0.01	-7.22 ± 0.00	-8.71 ± 0.02	-7.21 ± 0.01
β_N Sim / Real	$\mathbf{-16.41 \pm 0.30}$	-31.21 ± 1.67	-32.19 ± 1.19	-31.49 ± 3.66	-43.78 ± 6.46	-26.83 ± 1.36
β_N-Rotation Sim / Sim	-27.56 ± 0.08	-18.53 ± 0.02	-18.61 ± 0.12	-18.79 ± 0.09	-21.36 ± 0.05	$\mathbf{-18.45 \pm 0.03}$
β_N-Rotation Sim / Real	$\mathbf{-30.08 \pm 0.95}$	-45.91 ± 2.10	-44.24 ± 1.33	-48.23 ± 0.25	-60.23 ± 3.20	-41.86 ± 0.69
Average	-18.33	-20.12	-21.14	-18.71	-19.58	-17.51

Table 25: **Tracking Experiments with GRU+VIB**. We use green and red text to highlight significant improvements and deteriorations in performance over vanilla GRU. We only highlight a subset of configurations since we are focused on the robustness properties. This table shows average (unnormalized) returns.

E Computation Details

We used an internal cluster of machines to run these experiments. We mostly leveraged Nvidia Titan X GPUs for this, but also used a few Nvidia GTX 1080s. It is difficult to get an accurate estimate of run time since job loads vary drastically on our cluster from other users. However, to train a single policy on DMSD to completion (1 million transitions collected, or 1,000 epochs) using PIDE takes roughly 4.5 hours, using GPIDE takes roughly 17.25 hours, using a GRU takes roughly 14.5 hours, and using a transformer takes roughly 21 hours. This is similar for other tracking tasks. For PyBullet tasks, using GPIDE took roughly 43.2 hours and using a transformer took roughly 64.2 hours. We note that our implementation of GPIDE is somewhat naive and could be vastly improved.

In particular, for exponential smoothing and summation heads, w_t can be cached to save on compute, which is not being done currently. This is a big advantage in efficiency that GPIDE (especially one without attention heads) has over transformers.