
Adapt On-the-Go: Behavior Modulation for Single-Life Robot Deployment

Anonymous Author(s)

Affiliation

Address

email

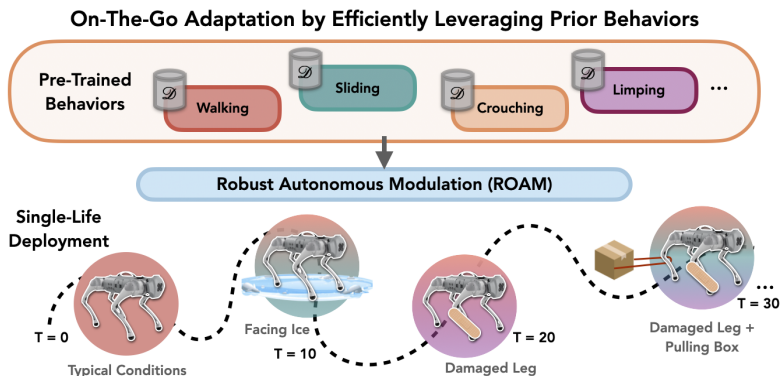


Figure 1: **On-The-Go Adaptation via Robust Autonomous Modulation (ROAM)**. An agent will inevitably encounter a wide variety of situations during deployment, and handling such situations may require a variety of different behaviors. We propose Robust Autonomous Modulation (ROAM), which dynamically employs and adapts relevant behaviors as different situations arise during a single trial of deployment.

1 Introduction

A major obstacle to the broad application of sequential decision-making agents is their inability to adapt to unexpected circumstances, which limits their uses largely to tightly controlled environments. Even equipped with prior experience and pre-training, agents will inevitably encounter out-of-distribution (OOD) situations at deployment time that may require a large amount of on-the-fly adaptation. In this work, we aim to enable a robot to autonomously handle novel scenarios encountered during deployment, while drawing upon a diverse set of pre-trained behaviors that may improve its versatility. We hypothesize that by doing some of the adaptation in the space of pre-trained behaviors (rather than only in the space of parameters), we can much more quickly react and adapt to novel circumstances. Accumulating a set of different behavior policies is a relatively straightforward task through online or offline episodic reinforcement learning, using different reward functions or skill discovery methods. However, the on-the-fly selection and adaptation of these behaviors during deployment, particularly in novel environments, present significant challenges. Consider tasking a quadrupedal robot that has acquired many different behaviors, e.g., walking, crouching, and limping via training in simulation, with a search-and-rescue mission in the real world. When deployed on this task with unstructured obstacles, the robot may bump into an obstacle and damage its leg, and it must be able to dynamically adapt its choice of behaviors to continue its mission with the damage.

Existing adaptation methods often operate within the standard, episodic training paradigm where the agent is assumed to be reset and have another chance to attempt the task each time (Cully et al., 2015; Song et al., 2020; Julian et al., 2020). However, these are idealized conditions that in practice often rely on human intervention. In the above search-and-rescue example, the robot’s state cannot be arbitrarily restored; during deployment, the robot cannot be repaired, and it is not feasible for a human

23 to fetch it if it gets stuck in a situation it is not equipped to handle. This situation necessitates adapting
24 at test time both quickly and autonomously, to succeed at the task within a single episode. Therefore,
25 we frame our problem setting as an instantiation of single-life deployment (Chen et al., 2022), where
26 the agent is given prior knowledge from ‘past lives’ but evaluated on its ability to successfully
27 complete a given task later in a ‘single-life’ trial, during which there are no human-provided resets.
28 The robot is provided with a diverse set of prior behaviors trained through episodic RL, and the single
29 life poses a sequence of new situations.

30 To solve this problem, the robot must identify during deployment which behaviors are most suited to
31 its situation at a given timestep and have the ability to fine-tune those behaviors in real time, as the
32 pre-trained behaviors may not optimally accommodate new challenges. We introduce a simple method
33 called RObust Autonomous Modulation (ROAM), which foremost aims to quickly identify the most
34 appropriate behavior from its pre-trained set at each point in time during single-life deployment.
35 Rather than introducing an additional component like a high-level controller to select behaviors,
36 we leverage the value function of each behavior. The value function already estimates whether
37 each policy will be successful, but may not be accurate for states that were not encountered during
38 training of that behavior. Therefore, prior to deployment, we fine-tune each behavior’s value function
39 with a regularized objective that encourages behavior identifiability: the regularizer is a behavior
40 classification loss. Then, at each step during deployment, ROAM samples a behavior proportional
41 to its classification probability, executes an action from that behavior, and optionally fine-tunes the
42 selected behavior for additional adaptation.

43 The main contribution of this paper is a simple algorithm for autonomous, deployment-time adaptation
44 to novel scenarios. In our theoretical analysis, we show that at a given state, with the additional
45 cross-entropy regularizer, ROAM can constrain each behavior’s value to be lower than the value of
46 behaviors for which that state appears more frequently. Consequently, our method incentivizes each
47 behaviors to differentiate between familiar and unfamiliar states, allowing ROAM to better recognize
48 when a behavior will be useful. We conduct experiments on both simulated locomotion tasks and on
49 a real Go1 quadruped robot. In simulation, our method completes the deployment task more than
50 two times faster on average than existing methods, including two prior methods designed for fast
51 adaptation. We also empirically analyze how the additional cross-entropy term in the loss function
52 of ROAM contributes to more successful utilization of the prior behaviors. Furthermore, ROAM
53 enables the Go1 robot to adapt on-the-go to various OOD situations without human interventions or
54 supervision in the real world. With ROAM, the robot can successfully pull heavy luggage, pull loads
55 with dynamic weights, and even slide forward with two roller skates on its front feet, even though it
56 never encountered loads or wore roller skates during training.

57 **2 Robust Autonomous Modulation**

58 We now present our method, Robust Autonomous Modulation (ROAM), which fine-tunes value
59 functions with an additional loss and provides a mechanism for choosing among them, so that at
60 deployment time, the agent can quickly react to its current situation *at every timestep* by honing
61 in on the most appropriate behavior from its prior behaviors. Our key observation is that with
62 proper regularization, value functions provide a good indication of how well different behaviors
63 will perform, so we can leverage them to quickly identify appropriate behaviors in a given situation,
64 which circumvents the need to learn a separate meta-controller or adaptation module.

65 **Fine-Tuning Value Functions with ROAM.** We desire value functions that accurately reflect the
66 expected reward of using a behavior at a given state. Thus, we would like to fine-tune the value
67 functions of the behaviors to minimize overestimation of the values at states for which a different
68 behavior is more suitable. We can do so by incentivizing the value functions of the behaviors to be
69 higher for states that are visited by that behavior and lower for states visited by other behaviors.

70 Given a set of prior behaviors \mathcal{B} with policies π_i and critics Q_i , we first fine-tune the value functions
71 of the behaviors with an additional cross-entropy loss on top of the Bellman error that takes in the
72 values at a given state as the logits. Values at a given state s for behavior i are obtained by averaging
73 $Q_i(s, a)$ over $N = 5$ sampled actions $a \sim \pi_i(\cdot | s)$. More formally, with each prior data buffer \mathcal{D}_i ,

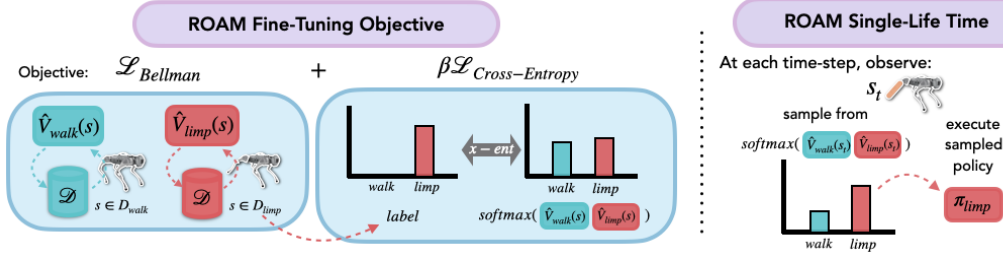


Figure 2: **Robust Autonomous Modulation (ROAM)**. During the initial fine-tuning phase of ROAM, we fine-tune each behavior using its existing data buffer and standard Bellman error, with an additional cross-entropy loss between the softmax values of all behaviors and the behavior index (as the label) from which the state was visited. Then at test-time, at each time-step, we sample from the softmax distribution of the behaviors’ values at the current state and execute the sampled policy.

74 we fine-tune the critic $Q_i(s, a)$ of each behavior i with the following update:

$$\begin{aligned}
 \mathcal{L}_{\text{fine-tune}} &= \mathcal{L}_{\text{Bellman}} + \beta \mathcal{L}_{\text{cross-entropy}} \\
 &= \sum_i \mathbb{E}_{s, a, s' \sim \mathcal{D}_i} \left[\left(r(s, a) + \gamma \mathbb{E}_{a' \sim \pi_i(a'|s')} Q_i(s', a') - Q_i(s, a) \right)^2 \right] \\
 &\quad + \beta \sum_j \mathbb{E}_{s \sim \mathcal{D}_j} \left[-\log \frac{\exp(V_j(s))}{\sum_{k=1}^n \exp(V_k(s))} \right],
 \end{aligned} \tag{1}$$

75 where β is a hyperparameter, $V_i(s) = \mathbb{E}_{a \sim \pi_i(a|s)} [Q_i(s, a)]$ is the average value of behavior i at state
 76 s , and \mathcal{D}_i is a replay buffer collected by behavior i . Consider the derivative of the cross-entropy term
 77 with respect to the value functions $V_i(s)$ and $V_k(s)$, where $i \neq k$, for a state s visited by behavior i :

$$\frac{\partial \mathcal{L}_{\text{cross-entropy}}}{\partial V_i(s)} = \frac{\exp(V_i(s))}{\sum_{j=1}^n \exp(V_j(s))} - 1 < 0, \quad \frac{\partial \mathcal{L}_{\text{cross-entropy}}}{\partial V_k(s)} = \frac{\exp(V_k(s))}{\sum_{j=1}^n \exp(V_j(s))} > 0.$$

78 So when minimizing the cross-entropy loss, the value function $V_i(s)$ will be pushed up (since its
 79 derivative is negative), and $V_k(s)$ for $k \neq i$ will be pushed down. Thus, the cross-entropy loss term in
 80 Equation 1 pushes up the value functions of the behaviors for states that are visited by that behavior
 81 and pushes down for states that are visited by other behaviors. The value functions are then less likely
 82 to overestimate at OOD states, enabling the behaviors to specialize in different parts of the state
 83 space, which will help us at test time to better infer an appropriate behavior from the current state.

84 **Full Procedure and Single-Life Deployment.** To summarize the full procedure of ROAM, we
 85 are given a set of policies π_i and critics Q_i , and a set of prior data buffers \mathcal{D}_i for each behavior.
 86 Alternatively, this can be relaxed and we can assume that we are given a set of prior data buffers \mathcal{D}_i
 87 for each behavior, and we can train the policies π_i and critics Q_i using these buffers with offline RL.
 88 We then fine-tune the value functions of the behaviors with the additional cross-entropy loss term in
 89 Equation 1 to obtain the final value functions V_i .

90 Then during each timestep at test time, we sample a behavior from the softmax distribution given
 91 by the behaviors’ values V_i at the current state. Formally, given the current state s , we sample an
 92 action $a_i \sim \pi_i(a|s)$ from behavior i with probability proportional to $\exp(V_i(s))$. The transition
 93 (s_t, a_t, r_t, s_{t+1}) is then added to the online buffer $\mathcal{D}_{\text{online}}^i$ for behavior i and the critic V_i and policy
 94 π_i are fine-tuned using data from $\mathcal{D}_{\text{online}}^i$. In this manner, we can choose and adapt the most suitable
 95 behavior for a given state on-the-fly. The ROAM fine-tuning objective and single-life deployment
 96 are depicted in Figure 2 and the full algorithm is summarized in Algorithms 1 and 2 in Appendix F.
 97 Theoretical results analyzing our method are given in Appendix D.

98 3 Experimental Results

99 In this section, we evaluate the performance of ROAM empirically and assess how effectively it can
 100 adapt on-the-fly to new situations. For qualitative video results, see our anonymous project webpage:
 101 <https://sites.google.com/view/adapt-on-the-go/home>. Our agent setup builds on top of Smith
 102 et al. (2022), and we include all experiment setup details and hyperparameters in Appendix H.

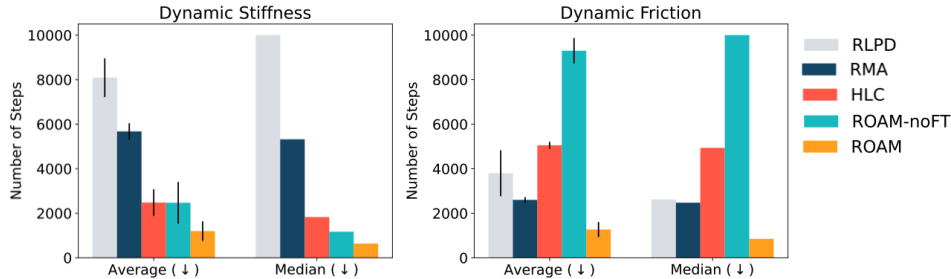


Figure 3: **Results on the simulated Go1 robot.** In both evaluation settings, ROAM is over 2x as efficient as all comparisons in terms of both average and median number of steps taken to complete the task.

	Heavy Luggage		Dynamic Luggage Load		Roller Skates	
	Avg. Time (s) ↓	Falls ↓	Avg. Time (s) ↓	Falls ↓	Avg. Time (s) ↓	Falls ↓
Walking	45.3	2.3	32	1	NC	NC
HLC	42.7	3	28.3	1.3	62.3	2.7
ROAM (ours)	25.7	0.7	24.3	0.3	27.3	1

Table 1: **Results on the real Go1 robot on 3 different tasks:** On all 3 tasks, across 3 trials for each method, ROAM significantly outperforms both comparisons in terms of both average wall clock time (s) and number of falls or readjustments needed to complete the task in a single life. NC (no complete) indicates that the task was not able to be successfully completed at all with the given method.

103 **Comparisons.** We evaluate ROAM along with the following prior methods: (1) RLPD Fine-tuning
 104 (Ball et al., 2023), where we fine-tune a single policy using all the data from the prior behaviors
 105 with RLPD; (2) RMA (Kumar et al., 2021), which trains a base policy and adaptation module that
 106 estimates environment info; (3) High-level Classifier (HLC), which trains a classifier on the data
 107 buffers of the pre-trained behaviors and uses it to select which behavior to use at a given state, as a
 108 representative method for those that train an additional behavior selection network, similar to work by
 109 Han et al. (2023). We additionally consider an ablation, ROAM-NoFT, which uses values to choose
 110 among behaviors but does not fine-tune with the additional cross-entropy loss.

111 **Selecting Relevant Behaviors in Simulation.** As seen in Figure 3, ROAM outperforms all other
 112 methods in all three metrics of average and median number of steps taken to complete the task as
 113 well as overall success rate. In particular, in both settings, ROAM completes the task *more than*
 114 *2 times faster*, in terms of average number of timesteps, compared to the next best method. Both
 115 RLPD fine-tuning and RMA struggle on both evaluation settings, especially the stiffness setting,
 116 demonstrating the importance of adapting in the space of behaviors rather than the space of actions
 117 for more efficient adaptation. On the other hand, HLC and ROAM-NoFT both struggle in the friction
 118 eval suite, demonstrating the importance of the additional cross-entropy term in the loss function,
 119 encouraging greater behavior specialization in different regions of the state space. These two methods
 120 perform better when the behaviors are already more distinguished, as in the limping eval suite, but
 121 they still struggle to adapt as efficiently as ROAM.

122 **Adapting on-the-Go1 in the Real World.** Although none of the prior behaviors are trained to handle
 123 these specific test-time scenarios, the robot can leverage parts of the prior behaviors to complete the
 124 task. As shown in Table 1, ROAM significantly outperforms using a high-level classifier (HLC) as
 125 well as the baseline walking policy in terms of both average wall clock time and number of falls or
 126 readjustments at single-life time on all three real-world tasks. Qualitatively, the other methods have
 127 trouble pulling luggage consistently forward, whereas our method often chooses the behavior where
 128 a joint is frozen on the leg with the luggage attached, as this behavior uses the robot’s other three legs
 129 to pull itself forward more effectively. The other methods struggle particularly on the roller skates
 130 task, which has drastically different dynamics from typical walking and especially relies on choosing
 131 relevant behaviors that heavily use the back legs. As seen in Figure 6, for all three tasks, HLC and
 132 the standard walking policy often fall or need to be readjusted very early in each single-life trial,
 133 whereas ROAM gets much closer to the finish line and often even completes the task without any
 134 falls or readjustments.

References

- 135
136 Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery
137 algorithms. *arXiv preprint arXiv:1807.10299*, 2018.
- 138 Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in chal-
139 lenging terrains using egocentric vision. In *Conference on Robot Learning, 2022*.
- 140 Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron,
141 Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik’s cube with a
142 robot hand. *arXiv preprint arXiv:1910.07113*, 2019.
- 143 Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint*
144 *arXiv:1607.06450*, 2016.
- 145 Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of*
146 *the AAAI conference on artificial intelligence*, volume 31, 2017.
- 147 Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning
148 with offline data. *arXiv preprint arXiv:2302.02948*, 2023.
- 149 Kate Baumli, David Warde-Farley, Steven Hansen, and Volodymyr Mnih. Relative variational
150 intrinsic control. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp.
151 6732–6740, 2021.
- 152 Annie Chen, Archit Sharma, Sergey Levine, and Chelsea Finn. You only live once: Single-life
153 reinforcement learning. *Advances in Neural Information Processing Systems*, 35:14784–14797,
154 2022.
- 155 Rohan Chitnis, Shubham Tulsiani, Saurabh Gupta, and Abhinav Gupta. Efficient bimanual manip-
156 ulation using learned task schemas. In *2020 IEEE International Conference on Robotics and*
157 *Automation (ICRA)*, pp. 1149–1155. IEEE, 2020.
- 158 Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like
159 animals. *Nature*, 521(7553):503–507, 2015.
- 160 Mark Cutler, Thomas J Walsh, and Jonathan P How. Reinforcement learning with multi-fidelity
161 simulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp.
162 3888–3895. IEEE, 2014.
- 163 Murtaza Dalal, Deepak Pathak, and Russ R Salakhutdinov. Accelerating robotic reinforcement
164 learning via parameterized action primitives. *Advances in Neural Information Processing Systems*,
165 34:21847–21859, 2021.
- 166 Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. **RI²**: Fast
167 reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.
- 168 Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need:
169 Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.
- 170 Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov.
171 Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint*
172 *arXiv:2006.13916*, 2020.
- 173 Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of
174 deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.
- 175 Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: learning a unified policy
176 for manipulation and locomotion. In *Conference on Robot Learning, 2022*.

- 177 Bolin Gao and Laca Pavel. On the properties of the softmax function with application in game theory
178 and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.
- 179 Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv*
180 *preprint arXiv:1611.07507*, 2016.
- 181 Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy
182 maximum entropy deep reinforcement learning with a stochastic actor. In *International conference*
183 *on machine learning*, pp. 1861–1870. PMLR, 2018.
- 184 Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H Huang, Dhruva Tirumala, Markus Wulfmeier,
185 Jan Humplik, Saran Tunyasuvunakool, Noah Y Siegel, Roland Hafner, et al. Learning agile soccer
186 skills for a bipedal robot with deep reinforcement learning. *arXiv preprint arXiv:2304.13653*,
187 2023.
- 188 Lei Han, Qingxu Zhu, Jiapeng Sheng, Chong Zhang, Tingguang Li, Yizheng Zhang, He Zhang,
189 Yuzhen Liu, Cheng Zhou, Rui Zhao, et al. Lifelike agility and play on quadrupedal robots using
190 reinforcement learning and generative pre-trained models. *arXiv preprint arXiv:2308.15143*, 2023.
- 191 Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel
192 Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint*
193 *arXiv:2007.04309*, 2020.
- 194 Rein Houthoofd, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, OpenAI Jonathan Ho, and
195 Pieter Abbeel. Evolved policy gradients. *Advances in Neural Information Processing Systems*, 31,
196 2018.
- 197 Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a
198 control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and*
199 *Automation Letters*, 2022.
- 200 Ryan Julian, Benjamin Swanson, Gaurav Sukhatme, Sergey Levine, Chelsea Finn, and Karol Haus-
201 man. Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning. 2020.
202 URL <https://arxiv.org/abs/2004.10190>.
- 203 Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement
204 learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.
- 205 Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for
206 legged robots. *arXiv preprint arXiv:2107.04034*, 2021.
- 207 Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Cic:
208 Contrastive intrinsic control for unsupervised skill discovery. *arXiv preprint arXiv:2202.00161*,
209 2022.
- 210 Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning
211 quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
- 212 Youngwoon Lee, Jingyun Yang, and Joseph J Lim. Learning to coordinate manipulation skills via
213 skill behavior diversification. In *International conference on learning representations*, 2019.
- 214 Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via
215 reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.
- 216 Russell Mendonca, Xinyang Geng, Chelsea Finn, and Sergey Levine. Meta-reinforcement learning
217 robust to distributional shift via model identification and experience relabeling. *arXiv preprint*
218 *arXiv:2006.07178*, 2020.
- 219 Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter.
220 Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 2022.

- 221 Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning
222 for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018a.
- 223 Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical
224 reinforcement learning. *Advances in neural information processing systems*, 31, 2018b.
- 225 Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and
226 Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement
227 learning. *arXiv preprint arXiv:1803.11347*, 2018.
- 228 Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting reinforcement learning with behavior
229 primitives for diverse manipulation tasks. In *2022 International Conference on Robotics and
230 Automation (ICRA)*, pp. 7477–7484. IEEE, 2022.
- 231 Seohong Park and Sergey Levine. Predictable mdp abstraction for unsupervised model-based rl.
232 *arXiv preprint arXiv:2302.03921*, 2023.
- 233 Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of
234 robotic control with dynamics randomization. In *2018 IEEE international conference on robotics
235 and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.
- 236 Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning
237 composable hierarchical control with multiplicative compositional policies. *Advances in Neural
238 Information Processing Systems*, 32, 2019.
- 239 Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine.
240 Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*,
241 2020.
- 242 Karl Pertsch, Youngwoon Lee, Yue Wu, and Joseph J Lim. Guided reinforcement learning with
243 learned skills. *arXiv preprint arXiv:2107.10253*, 2021.
- 244 Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning
245 robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.
- 246 Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal
247 meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.
- 248 Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray
249 Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint
250 arXiv:1606.04671*, 2016.
- 251 Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osin-
252 dero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint
253 arXiv:1807.05960*, 2018.
- 254 Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image.
255 *arXiv preprint arXiv:1611.04201*, 2016.
- 256 Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware
257 unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.
- 258 Mohit Sharma, Jacky Liang, Jialiang Zhao, Alex LaGrassa, and Oliver Kroemer. Learning to compose
259 hierarchical object-centric controllers for robotic manipulation. *arXiv preprint arXiv:2011.04627*,
260 2020.
- 261 Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes
262 with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

- 263 Xingyou Song, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn,
264 and Jie Tan. Rapidly adaptable legged robots via evolutionary meta-learning. In *2020 IEEE/RSJ*
265 *International Conference on Intelligent Robots and Systems (IROS)*, pp. 3769–3776. IEEE, 2020.
- 266 Robin Strudel, Alexander Pashevich, Igor Kalevtykh, Ivan Laptev, Josef Sivic, and Cordelia Schmid.
267 Learning to combine primitive skills: A step towards versatile robotic manipulation S. In *2020*
268 *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4637–4643. IEEE, 2020.
- 269 Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and
270 Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint*
271 *arXiv:1804.10332*, 2018.
- 272 Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain
273 randomization for transferring deep neural networks from simulation to the real world. In *2017*
274 *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE,
275 2017.
- 276 Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom
277 Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for
278 continuous control. *Software Impacts*, 6:100022, 2020.
- 279 Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos,
280 Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv*
281 *preprint arXiv:1611.05763*, 2016.
- 282 Annie Xie and Chelsea Finn. Lifelong robotic reinforcement learning by retaining experiences. *arXiv*
283 *preprint arXiv:2109.09180*, 2021.
- 284 Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst lifelong non-
285 stationarity. *arXiv preprint arXiv:2006.10701*, 2020.
- 286 Zhaoming Xie, Xingye Da, Michiel Van de Panne, Buck Babich, and Animesh Garg. Dynamics
287 randomization revisited: A case study for quadrupedal locomotion. In *2021 IEEE International*
288 *Conference on Robotics and Automation (ICRA)*, pp. 4955–4961. IEEE, 2021.
- 289 Ruihan Yang, Ge Yang, and Xiaolong Wang. Neural volumetric memory for visual locomotion
290 control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*,
291 2023.
- 292 Takuma Yoneda, Ge Yang, Matthew R Walter, and Bradley Stadie. Invariance through inference. *arXiv*
293 *preprint arXiv:2112.08526*, 2021.
- 294 Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal
295 policy with online system identification. *RSS*, 2017.
- 296 Wenhao Yu, Visak CV Kumar, Greg Turk, and C Karen Liu. Sim-to-real transfer for biped locomotion.
297 In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3503–3510.
298 IEEE, 2019.
- 299 Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with
300 meta strategy optimization. *IEEE Robotics and Automation Letters*, 2020.
- 301 Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher G Atkeson, Sören Schwertfeger, Chelsea Finn,
302 and Hang Zhao. Robot parkour learning. In *7th Annual Conference on Robot Learning*, 2023.

303 **A abstract**

304 To succeed in the real world, robots must cope with situations that differ from those seen during
305 training. We study the problem of adapting on-the-fly to such novel scenarios during deployment,
306 by drawing upon a diverse repertoire of previously-learned behaviors. Our approach, ROust
307 Autonomous Modulation (ROAM), introduces a mechanism based on the perceived value of pre-
308 trained behaviors to select and adapt pre-trained behaviors to the situation at hand. Crucially, this
309 adaptation process all happens within a single episode at test time, without any human supervision.
310 We provide theoretical analysis of our selection mechanism and demonstrate that ROAM enables
311 a robot to adapt rapidly to changes in dynamics both in simulation and on a real Go1 quadruped,
312 even successfully moving forward with roller skates on its feet. Our approach adapts over 2x as
313 efficiently compared to existing methods when facing a variety of out-of-distribution situations during
314 deployment by effectively choosing and adapting relevant behaviors on-the-fly.

315 **B Related Work**

316 We consider the problem of enabling an agent to act robustly when transferring to unstructured
317 test-time conditions that are unknown at train-time. One common instantiation of this problem is
318 transfer to different dynamics, e.g., in order to transfer policies trained in simulation to the real
319 world. A popular approach in achieving transfer under dynamics shift is domain randomization, i.e.,
320 randomizing the dynamics during training (Cutler et al., 2014; Rajeswaran et al., 2016; Sadeghi &
321 Levine, 2016; Tobin et al., 2017; Peng et al., 2018; Tan et al., 2018; Yu et al., 2019; Akkaya et al.,
322 2019; Xie et al., 2021; Margolis et al., 2022; Haarnoja et al., 2023) to learn a robust policy. Our
323 approach is similar in that it takes advantage of different MDPs during training; however, a key
324 component of ROAM is to leverage and modulate diverse skills rather than a single, robust policy.
325 We find in Section 3 that challenging test-time scenarios may require distinctly different behaviors at
326 different times, and we design our method to be robust to those heterogeneous conditions.

327 A class of methods that use domain randomization has also utilized different ‘behaviors’ instead
328 of a one-size-fits-all policy. Especially effective in locomotion applications, these methods involve
329 training policies that exhibit different behavior when conditioned on dynamics parameters, then
330 distilling these policies into one that can be deployed in target domains where this information is not
331 directly observable. The train-time supervision can come in the form of the parameter values (Yu
332 et al., 2017; Ji et al., 2022) or a learned representation of them (Lee et al., 2020; Kumar et al., 2021).
333 Thereafter, there are several ways prior work have explored utilizing online data to identify which
334 behavior is appropriate on-the-fly, e.g., using search in latent space (Yu et al., 2019; Peng et al., 2020;
335 Yu et al., 2020), or direct inference using proprioceptive history (Lee et al., 2020; Kumar et al., 2021;
336 Fu et al., 2022), or prediction based on egocentric depth (Miki et al., 2022; Agarwal et al., 2022;
337 Zhuang et al., 2023; Yang et al., 2023). In this work, we do not rely on domain-specific information
338 nor external supervision for when particular pre-trained behaviors are useful. Moreover, in contrast
339 to many of the above works, we focus on solving tasks that may be OOD for all prior behaviors
340 individually. By re-evaluating and potentially switching behaviors at every timestep, ROAM can take
341 the most useful parts of different behaviors to solve such tasks, as we find in Section 3.

342 Meta-RL is another line of work that achieves rapid adaptation without privileged information by
343 optimizing the adaptation procedure during training (Wang et al., 2016; Duan et al., 2016; Finn
344 et al., 2017; Nagabandi et al., 2018; Houthoofd et al., 2018; Rothfuss et al., 2018; Rusu et al., 2018;
345 Mendonca et al., 2020; Song et al., 2020) to be able to adapt quickly to a new situation at test-time.
346 Meta-RL and the aforementioned domain randomization-based approaches entangle the training
347 processes with the data collection, requiring a lot of *online* samples that are collected in a particular
348 way for pre-training. The key conceptual difference in our approach is that ROAM is *agnostic to*
349 *how the pre-trained policies and value functions are obtained*. Moreover, while meta-RL methods
350 often use hundreds of pre-training tasks, or more, our approach can provide improvements in new
351 situations even with a relatively small set of pre-trained behaviors (e.g. just 4 different behaviors
352 improve performance in Section 3). Other transfer learning approaches adapt the weights of the
353 policy to a new environment or task, either through rapid zero-shot adaptation (Hansen et al., 2020;

354 Yoneda et al., 2021; Chen et al., 2022) or through extended episodic online training (Khetarpal et al.,
 355 2020; Rusu et al., 2016; Eysenbach et al., 2020; Xie et al., 2020; Xie & Finn, 2021). Unlike these
 356 works, we focus on adaptation within a single episode to a variety of different situations.

357 Another rich body of work considers how to combine prior behaviors to solve long-horizon tasks,
 358 and some of these works also focus on learning or discovering useful skills (Gregor et al., 2016;
 359 Achiam et al., 2018; Eysenbach et al., 2018; Nachum et al., 2018a; Sharma et al., 2019; Baumli et al.,
 360 2021; Laskin et al., 2022; Park & Levine, 2023). Many of these methods involve training a high-level
 361 policy that learns to compose learned skills into long-horizon behaviors (Bacon et al., 2017; Peng
 362 et al., 2019; Lee et al., 2019; Sharma et al., 2020; Strudel et al., 2020; Nachum et al., 2018b; Chitnis
 363 et al., 2020; Pertsch et al., 2021; Dalal et al., 2021; Nasiriany et al., 2022). We show in Section 3 that
 364 training such a high-level policy is not needed for effective on-the-go behavior selection. Moreover,
 365 our work does not focus on where the behaviors come from – they could be produced by these prior
 366 methods, or their rewards could be specified manually. Instead, we focus on quickly selecting and
 367 adapting the most suitable skill in an OOD scenario, without requiring an additional online training
 368 phase to learn a hierarchical controller.

369 C Preliminaries

370 In this section, we describe some preliminaries and formalize our problem statement. We are given a
 371 set of n prior behaviors, where each behavior i is trained through episodic RL for a particular MDP
 372 $\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}_i, \mathcal{R}_i, \rho_0, \gamma)$ where \mathcal{S} is the state space, \mathcal{A} is the agent’s action space, $\mathcal{P}_i(s_{t+1}|s_t, a_t)$
 373 represents the environment’s transition dynamics, $\mathcal{R}_i : \mathcal{S} \rightarrow \mathbb{R}$ indicates the reward function,
 374 $\rho_0 : \mathcal{S} \rightarrow \mathbb{R}$ denotes the initial state distribution, and $\gamma \in [0, 1)$ denotes the discount factor. Each of
 375 the n MDPs \mathcal{M}_i has potentially different dynamics and reward functions \mathcal{P}_i and \mathcal{R}_i , often leading to
 376 different state visitation distributions. Each behavior corresponds to a policy π_i and a value function
 377 V_i as well as a buffer of trajectories $\tau \in \mathcal{D}_i$ collected during this prior training and relabeled with the
 378 reward $\mathcal{R}_{\text{target}}$ from the target MDP. At test time, the agent interacts with a target MDP defined by
 379 $\mathcal{M}_{\text{target}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_{\text{target}}, \mathcal{R}_{\text{target}}, \rho_0, \gamma)$, which presents an aspect of novelty not present in any of the
 380 prior MDPs, in the form of new dynamics $\mathcal{P}_{\text{target}}(s_{t+1} | s_t, a_t)$, which may *change over the course*
 381 *of the test-time trial*. We operate in a single-life deployment setting (Chen et al., 2022) that aims to
 382 maximize $J = \sum_{t=0}^h \gamma^t \mathcal{R}(s_t)$, where h is the trial horizon, which may be ∞ . The agent needs to
 383 complete the desired task in this target MDP in a single life without any additional supervision or
 384 human intervention by effectively selecting and adapting the prior behaviors to the situation at hand.

385 Off-policy reinforcement learning (RL) algorithms train a parametric Q-function, represented as
 386 $Q_\theta(s, a)$, via iterative applications of the Bellman optimality operator, expressed as

$$B^*Q(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(s'|s, a)} \left[\max_{a'} Q(s', a') \right].$$

387 In actor-critic frameworks, a separate policy is trained to maximize Q-values. These algorithms
 388 alternate between policy evaluation, which involves the Bellman operator $B^\pi Q = r + \gamma P^\pi Q$, and
 389 policy improvement, where the policy $\pi(a|s)$ is updated to maximize expected Q-values.

390 The dataset D_β , containing tuples (s, a, r, s') gathered by a behavior policy π_β , typically lacks full
 391 coverage of all possible transitions. Therefore, an empirical Bellman operator, denoted as \hat{B}^π , is used
 392 during policy evaluation. Specifically, the respective policy evaluation and improvement updates are:

$$Q^{k+1} \leftarrow \arg \min_Q \mathbb{E}_{s, a, s' \sim D} \left[\left(r(s, a) + \gamma \mathbb{E}_{a' \sim \pi^k(a'|s')} [Q^k(s', a')] - Q(s, a) \right)^2 \right]$$

393

$$\pi^{k+1} \leftarrow \arg \max_\pi \mathbb{E}_{s \sim D, a \sim \pi^k(a|s)} [Q^{k+1}(s, a)].$$

394 We use a state-of-the-art off-policy actor-critic algorithm RLPD (Ball et al., 2023) as our base
 395 algorithm for pre-training and fine-tuning, as it has been shown to be effective especially in the latter
 396 regime. RLPD builds on regularized soft actor-critic (SAC) (Haarnoja et al., 2018), using Layer
 397 Normalization (Ba et al., 2016) in particular as a key component.

398 D Theoretical Analysis

399 Next, we theoretically analyze ROAM to show that the additional cross-entropy loss in ROAM will
 400 lead to more suitable behaviors being chosen at each timestep. In particular, ROAM rescales the
 401 value functions of the behaviors so that they are less likely to overestimate in states that are out of
 402 distribution for that behavior. Our main result, given in Theorem D.2, is that with ROAM, for some
 403 weight $\beta > 0$ on the cross-entropy term, at a given state, ROAM constrains each behavior’s value to
 404 be lower than the value of behaviors for which that state appears more frequently. This theorem gives
 405 us *conservative generalization* by reducing value overestimation in unfamiliar states—specifically,
 406 if at least one behavior is familiar with the current state, our chosen behavior will not have much
 407 worse performance than its value function estimate. Full proofs of the statements in this section are
 408 presented in Appendix E.

409 Notationally, let behavior i be associated with a reward $R_i(s) = \mathbb{E}_{a \sim \pi_i(\cdot|s)}[R_i(s, a)]$ and dynamics
 410 function P_i for policy π_i . Our modified Bellman operator is $\hat{B}^\pi V = \left[\hat{B}^{\pi_i} V_i \right]_{i=1}^n$, where for each
 411 value function V_i ,

$$(\hat{B}^{\pi_i} V_i)(s) = \left\{ (1 - \beta) \left(R_i(s) + \gamma \sum_{s' \in S, a \in A} \pi_i(a|s) P_i(s'|s, a) V_i(s') \right) + \beta \left(\frac{\exp(\tau V_i(s))}{\sum_{k=1}^n \exp(\tau V_k(s))} \right) \right\}.$$

412 **Lemma D.1.** *There exists a temperature $\tau > 0$ for which our modified Bellman operator $\hat{B}^\pi V =$
 413 $\left[\hat{B}^{\pi_i} V_i \right]_{i=1}^n$ is a contraction under the \mathcal{L}^∞ norm.*

414 By Lemma D.1, for some temperature $\tau > 0$, and by the contraction mapping theorem, our modified
 415 Bellman operator will converge to a fixed point. In the following theorem, we characterize this fixed
 416 point and use it to analyze how ROAM will adjust value estimates based on degree of familiarity. Let
 417 $p_i(s)$ denote the state visitation probability for a behavior i at state s .

418 **Theorem D.2.** *For any state s that is out of distribution for behavior i and is in distribution for
 419 another behavior j , i.e. $p_i(s) \ll p_j(s)$, if $\beta > 0$ is chosen to be large enough, then the value
 420 of behavior i learned by ROAM will be bounded above compared to value of behavior j , i.e.,
 421 $V_i(s) \leq V_j(s)$.*

422 As a result, for any states s that are out of distribution for behavior i , if we choose β large enough,
 423 the value learned $V_i(s)$ will not overestimate the value compared to the behavior that is most familiar
 424 with that state. Thus, at each time step, if one or more behaviors are familiar with the current state,
 425 the performance of the chosen behavior will not be much worse than its value function estimate. In
 426 this manner, ROAM adjusts value estimates based on degree of familiarity, mitigating overestimation
 427 risks. The ability to adjust the β parameter offers a flexible framework to optimize for the behavior
 428 with the highest value at a given state, which will be at least as suitable as the most familiar behavior.

429 In the next section, we find empirically that after fine-tuning with the additional cross-entropy loss,
 430 ROAM is able to effectively select a relevant behavior for a given state on-the-fly, leading to robust
 431 and fast adaptation to OOD situations.

432 E Proofs for Theoretical Analysis

433 In this section, we present the full proofs for Section D. Following the same notation, let behavior i
 434 be associated with reward $R_i = \mathbb{E}_{a \in \pi_i} [R_i(s, a)]$ and dynamics function P_i . Our modified Bellman
 435 operator is $\hat{B}^\pi V = \left[\hat{B}^{\pi_i} V_i \right]_{i=1}^n$, where for each value function V_i , the modified Bellman operator is
 436 the following:

$$(\hat{B}^{\pi_i} V_i)(s) = \left\{ (1 - \beta) \left(R_i(s) + \gamma \sum_{s' \in S, a \in A} \pi_i(a|s) P_i(s'|s, a) V_i(s') \right) + \beta \left(\frac{\exp(\tau V_i(s))}{\sum_{k=1}^n \exp(\tau V_k(s))} \right) \right\}.$$

437 **Lemma E.1.** *There exists $\tau > 0$, our modified Bellman operator $\hat{B}^\pi V = \left[\hat{B}^{\pi_i} V_i \right]_{i=1}^n$ is a contraction
 438 under the \mathcal{L}^∞ norm.*

439 *Proof.* In order to show this, we first show that for any element V_i of the full vector of value functions
 440 V , $(\hat{B}^{\pi_i} V_i)(s)$ is a contraction. For all $s \in \mathcal{S}$ and any two V_i, V'_i value functions:

$$\begin{aligned} |(\hat{B}^{\pi} V_i)(s) - (\hat{B}^{\pi} V'_i)(s)| &= |(1 - \beta) \left(R_i(s) + \gamma \sum_{s' \in \mathcal{S}, a \in A} \pi_i(a|s) P_i(s'|s, a) V_i(s') \right) + \beta \left(\frac{\exp(\tau V_i(s))}{\sum_{k=1}^n \exp(\tau V_k(s))} \right) \\ &\quad - (1 - \beta) \left(R_i(s) + \gamma \sum_{s' \in \mathcal{S}, a \in A} \pi_i(a|s) P_i(s'|s, a) V'_i(s') \right) - \beta \left(\frac{\exp(\tau V'_i(s))}{\sum_{k=1}^n \exp(\tau V'_k(s))} \right)| \\ &= |(1 - \beta) \left(\gamma \left(\sum_{s' \in \mathcal{S}, a \in A} \pi_i(a|s) P_i(s'|s, a) (V_i(s') - V'_i(s')) \right) \right) \\ &\quad + \beta \left(\frac{\exp(\tau V_i(s))}{\sum_{k=1}^n \exp(\tau V_k(s))} - \frac{\exp(\tau V'_i(s))}{\sum_{k=1}^n \exp(\tau V'_k(s))} \right)|. \end{aligned}$$

441 The first term becomes

$$\begin{aligned} |\mathbb{E}_{a \in A} (1 - \beta) \left(\gamma \left(\sum_{s' \in \mathcal{S}, a \in A} \pi_i(a|s) P_i(s'|s, a) (V_i(s') - V'_i(s')) \right) \right)| &\leq \gamma (1 - \beta) \max_{a \in A} \sum_{s' \in \mathcal{S}} P_i(s'|s, a) |V_i(s') - V'_i(s')| \\ &\leq \gamma (1 - \beta) \|V_i - V'_i\|_{\infty}. \end{aligned}$$

442 We now focus on the second term: Since the softmax function is Lipschitz continuous with respect to
 443 the L_2 norm (Gao & Pavel, 2017), for all $V, V' \in \mathbb{R}$, there exists $C > 0$ such that

$$\left\| \left(\frac{\exp(V_i)}{\sum_{k=1}^n \exp(V_k)} - \frac{\exp(V'_i)}{\sum_{k=1}^n \exp(V'_k)} \right) \right\|_2 \leq C \|V_i - V'_i\|_2 \leq C \sqrt{|S|} \|V_i - V'_i\|_{\infty}.$$

444 Let $0 < \tau < 1/(C\sqrt{|S|}) < \infty$. Then

$$\left\| \left(\frac{\exp(\tau V_i)}{\sum_{k=1}^n \exp(\tau V_k)} - \frac{\exp(\tau V'_i)}{\sum_{k=1}^n \exp(\tau V'_k)} \right) \right\|_{\infty} \leq C \sqrt{|S|} \|\tau V_i - \tau V'_i\|_{\infty} < \|V_i - V'_i\|_{\infty}.$$

445 Therefore, by the triangle inequality, for any $s \in \mathcal{S}$,

$$\begin{aligned} \|\hat{B}^{\pi} V_i - \hat{B}^{\pi} V'_i\|_{\infty} &= \max_s |\hat{B}^{\pi} V_i(s) - \hat{B}^{\pi} V'_i(s)| \\ &\leq \gamma (1 - \beta) \|V_i - V'_i\|_{\infty} + \beta \|V_i - V'_i\|_{\infty} \\ &< \|V_i - V'_i\|_{\infty}. \end{aligned}$$

446 Thus, considering our full modified Bellman operator, on n value functions V_1, \dots, V_n , we have

$$\begin{aligned} \|\hat{B}^{\pi} V - \hat{B}^{\pi} V'\|_{\infty} &= \max_i \max_s |\hat{B}^{\pi} V_i(s) - \hat{B}^{\pi} V'_i(s)| \\ &< \max_i \|V_i - V'_i\|_{\infty} \\ &< \|V - V'\|_{\infty}. \end{aligned}$$

447 □

448 **Theorem E.2.** Let $p_i(s)$ denote the state visitation probability for a behavior b_i at state s . For any
 449 state s that is out of distribution for behavior b_i and is in distribution for another behavior b_j , i.e.
 450 $p_i(s) \ll p_j(s)$, if $\beta > 0$ is chosen to be large enough, then the value of behavior i learned by ROAM
 451 will be bounded above compared to value of behavior b_j , i.e. $V_i(s) \leq V_j(s)$.

452 *Proof.* By Lemma E.1, for some temperature $\tau > 0$, by the contraction mapping theorem, our
 453 modified Bellman operator will lead to a fixed point. In the following, we characterize this fixed
 454 point and use it to analyze how ROAM will adjust value estimates based on degree of familiarity.

455 Our loss function is defined based on our modified Bellman operator, optimizing for the fixed point
 456 $\hat{B}^\pi V = V$, as follows:

$$\mathcal{L}_{\text{fine-tune}} = \sum_i \sum_{s \sim \mathcal{D}_i} p_i(s) \left[(R_i(s) + \gamma \mathbb{E}_{s'} V_i(s') - V_i(s))^2 \right] + \beta \sum_j \sum_{s \sim \mathcal{D}_j} p_j(s) \left[-V_j(s) + \log \sum_{k=1}^n \exp(V_k(s)) \right].$$

457 Taking the derivative with respect to $V_i(s)$, we have:

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{fine-tune}}}{\partial V_i(s)} &= 2p_i(s)(\gamma p(s|s, a) - 1) (R_i(s) + \gamma \mathbb{E}_{s'} V_i(s') - V_i(s)) \\ &\quad + \beta \left[p_i(s) \left(-1 + \frac{\exp(V_i(s))}{\sum_{k=1}^n \exp(V_k(s))} \right) + \sum_{j \neq i} p_j(s) \frac{\exp(V_i(s))}{\sum_{k=1}^n \exp(V_k(s))} \right] \\ &= 2p_i(s)(\gamma P_i(s|s, a) - 1) (R_i(s) + \gamma \mathbb{E}_{s'} V_i(s') - V_i(s)) + \beta \left[-p_i(s) + \sum_j p_j(s) \frac{\exp(V_i(s))}{\sum_{k=1}^n \exp(V_k(s))} \right]. \end{aligned} \quad .1$$

458 Setting to 0, we have the following characterization of our fixed point, obtained as a result of
 459 Theorem E.1 for some temperature $\tau > 0$:

$$\begin{aligned} V_i(s) &= (R_i(s) + \gamma \mathbb{E}_{s'} V_i(s')) + \frac{\beta}{2p_i(s)(\gamma P_i(s|s, a) - 1)} \left[-p_i(s) + \sum_j p_j(s) \frac{\exp(V_i(s))}{\sum_{k=1}^n \exp(V_k(s))} \right] \\ &= (R_i(s) + \gamma \mathbb{E}_{s'} V_i(s')) + \frac{\beta}{2(1 - \gamma P_i(s|s, a))} \left[1 - \sum_j \frac{p_j(s)}{p_i(s)} \frac{\exp(V_i(s))}{\sum_{k=1}^n \exp(V_k(s))} \right]. \end{aligned}$$

We examine the behavior of $V_i(s)$ under case $p_{\text{freq}}(s) \gg p_i(s)$ for some behavior $b_{\text{freq}} \neq b_i$. Then $\frac{p_{\text{freq}}(s)}{p_i(s)}$ will dominate in the last term, so

$$\sum_j \frac{p_j(s)}{p_i(s)} \frac{\exp(V_i(s))}{\sum_{k=1}^n \exp(V_k(s))} \gg 1.$$

460 Then comparing the fixed point values of $V_i(s)$ and $V_{\text{freq}}(s)$, we have

$$\begin{aligned} V_{\text{freq}}(s) - V_i(s) &= (R_{\text{freq}}(s) + \gamma \mathbb{E}_{s'} V_{\text{freq}}(s')) + \frac{\beta}{2(1 - \gamma P_{\text{freq}}(s|s, a))} \left[1 - \sum_j \frac{p_j(s)}{p_i(s)} \frac{\exp(V_{\text{freq}}(s))}{\sum_{k=1}^n \exp(V_k(s))} \right] \\ &\quad - \left[(R_i(s) + \gamma \mathbb{E}_{s'} V_i(s')) + \frac{\beta}{2(1 - \gamma P_i(s|s, a))} \left[1 - \sum_j \frac{p_j(s)}{p_i(s)} \frac{\exp(V_i(s))}{\sum_{k=1}^n \exp(V_k(s))} \right] \right] \\ &= ((R_{\text{freq}}(s) + \gamma \mathbb{E}_{s'} V_{\text{freq}}(s')) - (R_i(s) + \gamma \mathbb{E}_{s'} V_i(s'))) \\ &\quad + \beta \left(C_{\text{freq}} \left[1 - \sum_j \frac{p_j(s)}{p_i(s)} \frac{\exp(V_{\text{freq}}(s))}{\sum_{k=1}^n \exp(V_k(s))} \right] - C_i \left[1 - \sum_j \frac{p_j(s)}{p_i(s)} \frac{\exp(V_i(s))}{\sum_{k=1}^n \exp(V_k(s))} \right] \right) \\ &> 0, \end{aligned}$$

461 where $0 < C_{\text{freq}}, C_i < \infty$ constants, if

$$\beta > \frac{((R_i(s) + \gamma \mathbb{E}_{s'} V_i(s')) - (R_{\text{freq}}(s) + \gamma \mathbb{E}_{s'} V_{\text{freq}}(s')))}{\left(C_{\text{freq}} \left[1 - \sum_j \frac{p_j(s)}{p_i(s)} \frac{\exp(V_{\text{freq}}(s))}{\sum_{k=1}^n \exp(V_k(s))} \right] - C_i \left[1 - \sum_j \frac{p_j(s)}{p_i(s)} \frac{\exp(V_i(s))}{\sum_{k=1}^n \exp(V_k(s))} \right] \right)}.$$

462 Thus, for some $\beta > 0$ large enough, $V_i(s) < V_j(s)$. □

463 To illustrate the effect of the cross-entropy loss, consider the following example of choosing between
 464 two behaviors i and j at state s , where the true values $V_i^{\text{true}}(s) < V_j^{\text{true}}(s)$. The optimal choice is to
 465 choose behavior j and for sake of this example let us choose behavior j if $V_i^{\text{ROAM}}(s) < V_j^{\text{ROAM}}(s)$.
 466 There are the following four cases: (1) $p_i(s) < p_j(s)$ and the initial estimated $V_i(s) < V_j(s)$. Then
 467 with any $\beta > 0$, the final $V_i^{\text{ROAM}}(s) < V_j^{\text{ROAM}}(s)$; (2) $p_i(s) < p_j(s)$ and the initial estimated
 468 $V_i(s) > V_j(s)$. Then by Theorem E.2, with large enough $\beta > 0$, the final $V_i^{\text{ROAM}}(s) < V_j^{\text{ROAM}}(s)$;
 469 (3) $p_i(s) > p_j(s)$ and the initial estimated $V_i(s) < V_j(s)$. Then as long as β is chosen to be not too
 470 large, the final $V_i^{\text{ROAM}}(s) < V_j^{\text{ROAM}}(s)$. (4) $p_i(s) > p_j(s)$ and the initial estimated $V_i(s) > V_j(s)$.
 471 This is the only case where ROAM may not be adjusted to work well, but this case poses a difficult
 472 situation for any behavior selection method.

473 F Algorithm Summary

474 We summarize ROAM in Algorithms 1 and 2.

Algorithm 1 ROAM FINE-TUNING

```

1: Require:  $\mathcal{D}_i$ , pre-trained critics  $Q_i$ 
2: while not converged do
3:   for all  $i$  in  $1, \dots, N_{\text{behaviors}}$  do
4:     Sample  $(s, a, s', r) \sim \mathcal{D}_i$ 
5:     Update  $Q_i$  according to Eq. 1
6:   end for
7: end while
8: return  $Q_1, \dots, Q_{N_{\text{behaviors}}}$ 

```

Algorithm 2 ROAM SINGLE-LIFE DEPLOYMENT

```

1: Require: Test MDP  $\mathcal{M}_{\text{test}}$ ,  $\mathcal{D}_i$ , policies  $\pi_i$  and fine-
  tuned critics  $Q_i$ ;
2: Initialize: online replay buffers  $\mathcal{D}_{\text{online}}^i$ ; timestep  $t = 0$ 
3: while task not complete do
4:   Compute values of each behavior  $\{V_i(s_t)\}_1^{N_{\text{behaviors}}}$ 
5:   Sample behavior  $b^*$  according to the distribution
     softmax( $\exp(V_i(s_t))$ ).
6:   Take action  $a_t \sim \pi_{b^*}(a_t | s_t)$ .
7:    $\mathcal{D}_{\text{online}}^{b^*} \leftarrow \mathcal{D}_{\text{online}}^{b^*} \cup \{(s_t, a_t, r_t, s_{t+1})\}$ 
8:    $Q_{b^*}(s, a), \pi_{b^*} \leftarrow \text{RL}(Q_{b^*}(s, a), \pi_{b^*}, \mathcal{D}_{\text{online}}^{b^*})$ 
9:   Increment  $t$ 
10: end while

```

475 G Additional Experiment Results

476 In Figures 4 and 5, we provide some additional empirical analysis of ROAM. First, in the stiffness
 477 evaluation, we plot the percent of steps where different methods select a relevant behavior during
 478 test-time deployment, where a held-out joint is frozen and a relevant behavior is one where an adjacent
 479 joint on the same leg is frozen or the same joint on an adjacent leg is frozen, and we see that ROAM
 480 is able to choose the most relevant behavior on average significantly more frequently than HLC and
 481 ROAM-noFT, which often select a behavior that is not relevant to the current situation. In addition,
 482 we record the average values of the different behaviors in states visited by that behavior (ID) vs states
 483 visited by other behaviors (OOD), before and after fine-tuning with the additional cross-entropy
 484 term for ROAM. We find that ROAM is able to effectively maintain high values of the behavior in
 485 familiar states, while decreasing the value of the behavior in unfamiliar OOD states, leading to better
 486 distinction between relevant behaviors at a given state.

487 H General Experiment Setup Details

488 As common practice in learning-based quadrupedal locomotion works, we define actions to be PD
 489 position targets for the 12 joints, and we use a control frequency of 20 Hz. Actions are centered
 490 around the nominal pose, i.e. 0 is standing. We describe the observations for the simulated and
 491 real-world experiments below.

492 We first detail the reward function we use to define the quadrupedal walking task. First, we have a
 493 velocity-tracking term defined as follows:

$$r_v(s, a) = 1 - \left| \frac{v_x - v_t}{v_t} \right|^{1.6}$$

494 where v_t is the target velocity and v_x is the robot’s local, forward linear velocity projected onto the
 495 ground plane, i.e., $v_x = v_{\text{local}} \cdot \cos(\phi)$ where ϕ is the root body’s pitch angle. We then have a term

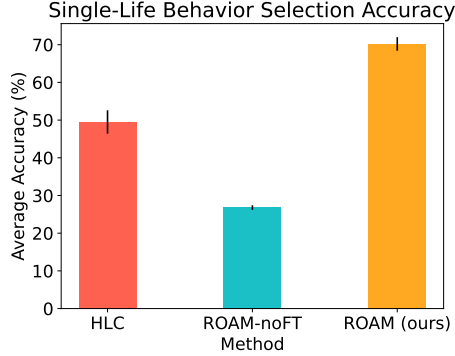


Figure 4: **Single-life Behavior Selection Accuracy.** The percent of steps where different methods select a relevant behavior for the current situation. ROAM is able to choose a relevant behavior significantly more often on average when adapting to test-time situations than HLC and ROAM-noFT.

		Before FT	After FT
Changing Stiffness	Avg ID Values	779.2	778.7
	Avg OOD Values	746.4	542.9
Changing Frictions	Avg ID Values	1243.4	1247.6
	Avg OOD Values	1217.6	1192.8

Figure 5: **Effect of the Cross-Entropy Loss on ID and OOD values.** The average values of the different behaviors in states visited by that behavior (ID) vs states visited by other behaviors (OOD), before and after fine-tuning with the additional cross-entropy term for ROAM. ROAM is able to maintain high values of the behavior in ID states, while decreasing the value of the behavior in OOD states, which leads to better distinction between relevant behaviors at a given state.

496 $r_{ori}(s, a)$ that encourages the robot to stay upright. Specifically, we calculate the cosine distance
 497 between the 3d vector perpendicular to the robot’s body and the gravity vector $([0, 0, 1])$. We then
 498 normalize the term to be between 0 and 1 via:

$$r_{ori}(s, a) = (0.5 \cdot \text{dist} + 0.5)^2$$

499 where dist is the cosine distance as described above. We multiply r_v and r_{ori} so as to give reward for
 500 tracking velocity proportionally to how well the robot is staying upright. We then have a regularization
 501 term r_{qpos} to favor solutions that are close to the robot’s nominal standing pose. This regularization
 502 term is calculated as a product of a normalized term per-joint. Below, \hat{q}^j represents the local rotation
 503 of joint j of the nominal pose, and q^j represents the robot’s joint,

$$r^{qpos} = 1 - \prod_j \text{q distance}(\hat{q}^j, q^j)$$

504 where q distance is between 0 and 1 and decays quadratically until a threshold which is the robot’s
 505 action limits. Specifically, we follow the reward structure put forth by Tunyasuvunakool et al. (2020).
 506 These terms comprise the overwhelming majority of weight in the final reward. We also include
 507 terms for avoiding undesirable behaviors like rocking or swaying that penalize any angular velocity
 508 in the root body’s roll, pitch, and yaw. We also slightly penalize energy consumption and torque
 509 smoothness. To encourage a walking gait in particular, we added another regularization term to
 510 encourage diagonal shoulder and hip joints to be the same at any given time.

511 **Simulation Exps Setup.** In our simulation experiments, we evaluate in two separate settings. The
 512 first setting simulates a situation where different joints become damaged or stuck during the robot’s
 513 lifetime. It uses 9 prior behaviors: each is a different limping behavior with a different joint frozen.
 514 In the single life, the agent must walk a total distance of 10 meters, and every 100 steps, one of the 3
 515 remaining joints *not covered in the prior data* is frozen, and the agent must adapt its walking behavior
 516 to continue walking. The second setting simulates a situation where the robot encounters different
 517 friction levels on its different feet due to variation in terrain. It uses 4 different prior behaviors, each
 518 of which is trained with one of the 4 feet having low friction. During the single life, every 50 steps,
 519 the friction of one or two of the feet is changed to be lower than in the prior behaviors. To collect
 520 the prior behaviors, we train each behavior for 250k steps (first setting) or 50k steps (second setting)
 521 in the corresponding MDP, and use the last 40k steps as the prior data. We report the average and
 522 median number of steps taken to complete the task across 10 seeds along with the standard error and
 523 success rate (out of 10) in Figure 3. The agent is given a maximum of 10k steps to complete the task,
 524 and if it does not complete the task within this time, it is considered an unsuccessful trial.

525 **Real Robot Exps Setup.** On the Go1 quadruped robot, we evaluate ROAM in a setting where we
 526 have a fixed set of five prior behaviors: walking, and four different behaviors where each of the legs

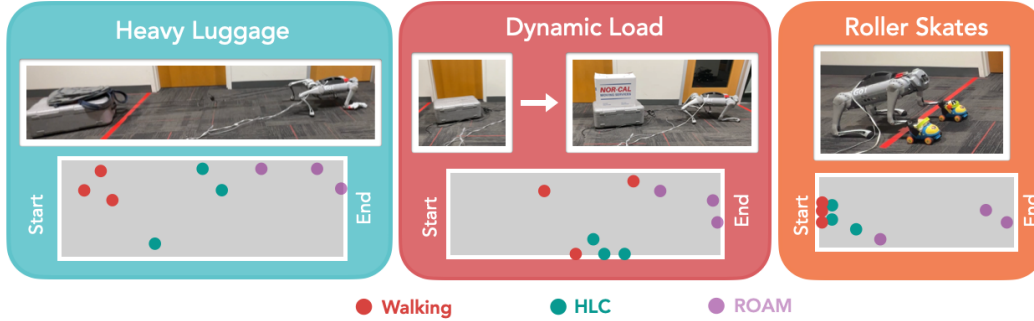


Figure 6: **Real-world single-life tasks.** We evaluate on: (1) pulling a load of heavy luggage 6.2 kg (13.6 lb), (2) pulling luggage where the weight dynamically changes between 2.36 kg (5.2 lb) and 4.2 kg (9.2 lb), and (3) moving forward with roller skates on the robot’s front two feet. For each trial, for each method, we also show the locations of the robot before its first fall or readjustment (Red is Walking, Blue is HLC, and Purple is ROAM).

527 has a joint frozen. We pre-train a base walking behavior in 18k steps and train the other behaviors by
 528 fine-tuning the walking behavior for an additional 3k steps with one of the joints frozen, all from
 529 scratch in the real world using the system from Smith et al. (2022). During single-life deployment, we
 530 evaluate on the following three tasks: (1) Heavy Luggage: the robot must walk from a starting line to
 531 a finish line, while pulling a box that is 6.2 kg (13.6 lb) attached to one of the back legs. In addition,
 532 one of the front legs is one leg is covered by a low-friction plastic protective layer and the robot has
 533 to figure out how to adapt to walk with this leg. (2) Dynamic Luggage Load: the robot must walk
 534 from a starting line to a finish line, while adapting on-the-fly to a varying amount of weight between
 535 2.36 kg (5.2 lb) and 4.2 kg (9.2 lb). We standardize each trial by adding and removing weight at the
 536 same distance from the start position. (3) Roller Skates: we fit the robot’s front two feet into roller
 537 skates, and the robot must adapt its behavior to slide its forward legs and push off its back legs in
 538 order to walk to the end line. for each of the three trials for each task. We report the average wall
 539 clock time in seconds and number of falls or readjustments needed to complete the task in a single
 540 life across 3 trials for each method. If the robot falls, a get-up controller (we use the open-sourced
 541 policy from Smith et al. (2022)) is triggered to reset the robot back to a standing position. If it walks
 542 into a wall, it is readjusted to face the correct direction. The tasks are shown in Figure 6, along with
 543 the locations of the robot before its first fall or readjustment for each method for each trial, where the
 544 red dots correspond to the Walking policy, blue to HLC, and purple to ROAM.

545 H.1 Implementation Details and Hyperparameters

546 We use RLPD (Ball et al., 2023) as our base algorithm for training the prior behaviors and for
 547 fine-tuning the behaviors during deployment. We use default hyperparameter values: a learning rate
 548 of 3×10^{-4} , an online batch size of 128, and a discount factor of 0.99. The policy and critic networks
 549 are MLPs with 2 hidden layers of 256 units each and ReLU activations. For ROAM, we tuned β with
 550 values 1, 10, 100, 1000.

551 **Simulated Experiments.** For the simulated experiments, the state space consists of joint positions,
 552 joint velocities, torques, IMU (roll, pitch, change in roll, change in pitch), and normalized foot
 553 forces for a total of 44 dimensions. For the position controller, we use K_p and K_d gains of 40 and 5,
 554 respectively, and calculate torques for linearly interpolated joint angles from current to desired at
 555 500Hz. We define the limits of the action space to be 30% of the physical joint limits.

556 For the first experimental setting, we train prior behavior policies with high stiffness (10.0) in 9
 557 different individual joints. Specifically, we use the front right body joint, the front right knee joint,
 558 the front left body joint, the front left knee joint, the rear right body joint, the rear right knee joint,
 559 the rear left body joint, the rear left thigh joint, and the rear left knee joint. During deployment,
 560 we switch between 3 conditions every 100 steps. Condition 1 is applying stiffness 15.0 to the rear
 561 right thigh joint, condition 2 is applying stiffness 15.0 to the front left thigh joint, and condition 3 is
 562 applying stiffness 15.0 to the front right thigh joint. For this setting, we use $\beta = 1$ for ROAM.

Table 2: Simulated Reward Function Parameter Details

Parameter	Value
Target Velocity	1.0
Energy Penalty Weight	0.008
Qpos Penalty Weight	10.0
Smooth Torque Penalty Weight	0.005
Pitch Rate Penalty Factor	0.6
Roll Rate Penalty Factor	0.6
Joint Diagonal Penalty Weight	0.1
Joint Shoulder Penalty Weight	0.15
Smooth Change in Target Delta Yaw Steps	5

563 For the second experimental setting, we train prior behavior policies with low foot friction (0.4) in
 564 each of the 4 feet. During deployment, we switch between 2 conditions every 50 steps. Condition 1
 565 is applying a foot friction of 0.1 to the rear right foot and condition 2 is applying a foot friction of
 566 0.01 to the front left foot and a foot friction of 0.1 too the rear right foot. For this setting, we use
 567 $\beta = 100$ for ROAM.

568 **Real-world Experiments.** For the real-world experiments, the state space consists of joint positions,
 569 joint velocities, torques, forward linear velocity, IMU (roll, pitch, change in roll, change in pitch),
 570 and normalized foot forces for a total of 47 dimensions. We use an Intel T265 camera-based velocity
 571 estimator to estimate onboard linear velocity. We use K_p and K_d gains of 20 and 1, respectively,
 572 which are used in the position controller. We again use action interpolation, an action range of 35%
 573 physical limits, and a 1 step action history. We also use a second-order Butterworth low-pass filter
 574 with a high-cut value of 8 to smooth the position targets. Finally, to reset the robot, we use the
 575 reset policy provided by Smith et al. (2022). We train 4 prior behavior policies for the real-world
 576 experiments, each of which is trained with a frozen knee joint. Specifically, we train a policy with the
 577 front right knee joint frozen, the front left knee joint frozen, the rear right knee joint frozen, and the
 578 rear left knee joint frozen. $\beta = 1$ is used in all real-world experiments for ROAM.

Table 3: Real-world Reward Function Parameter Details

Parameter	Value
Target Velocity	1.5
Energy Penalty Weight	0.0
Qpos Penalty Weight	2.0
Smooth Torque Penalty Weight	0.005
Pitch Rate Penalty Factor	0.4
Roll Rate Penalty Factor	0.2
Joint Diagonal Penalty Weight	0.03
Joint Shoulder Penalty Weight	0.0
Smooth Change in Target Delta Yaw Steps	1

579 **HLC Details.** For HLC, we have an MLP that takes state as input and outputs which behavior to
 580 select in the given state. The MLP has 3 hidden layers of 256 units each and ReLU activations, and
 581 we train by sampling from the combined offline data from all prior behaviors. We use a batch size of
 582 256, learning rate of 3×10^{-4} , and train for 3,000 iterations.

583 **RMA Details.** For RMA training, we changed the environment dynamics between each episode
 584 and trained for a total of 2,000,000 iterations. The standard architecture and hyperparameter choices
 585 from Kumar et al. (2021) were used along with DroQ as the base algorithm.

586 I Conclusion and Future Work

587 We introduced Robust Autonomous Modulation (ROAM), which enables agents to rapidly adapt to
 588 changing, out-of-distribution circumstances during deployment. Our contribution lies in offering

589 a principled, efficient way for agents to leverage pre-trained behaviors when adapting on-the-fly.
590 Through a value-based mechanism, ROAM identifies the most relevant pre-trained behaviors in
591 real-time at each time-step without any human supervision. Our theoretical analysis confirms the
592 effectiveness of our behavior modulation strategy, showing why a suitable behavior will be chosen for
593 a given state with ROAM. On simulated tasks and a complex real-world tasks with a Go1 quadruped
594 robot, we find that our method achieves over 2x efficiency in adapting to new situations compared
595 to existing methods. While ROAM offers significant advances in enabling agents to adapt to out-
596 of-distribution scenarios, one current limitation lies in the dependency on the range of pre-trained
597 behaviors; some scenarios may simply be too far out-of-distribution compared to the available prior
598 behaviors. For example, an agent trained primarily in walking tasks would struggle to adapt to
599 the requirement of jumping over an obstacle. Future work could explore integrating ROAM into a
600 lifelong learning framework, allowing agents to continuously expand their repertoire of behaviors,
601 thereby increasing their adaptability to more unforeseen situations. We hope that ROAM may open
602 new possibilities for more versatile and self-reliant autonomous systems.