Meta-Learning for Fast Adaptation in Assistive Navigation: A Capability-Aware Approach

Jasraj Budigam
Indus International School
India
jasrajharikrishna.b@indusschoolhyd.com

Abstract: Assistive navigation systems must adapt to the diverse capabilities of individual users, who may vary in motor control precision and device noise tolerance, and to different environmental layouts. This work presents a meta-learning approach that learns capability-aware priors using tabular Q-learning across a range of grid-based navigation tasks. The approach is evaluated in two regimes: Easy tasks on 8×8 grids with moderate clutter and no capability costs, and Hard tasks on 12×12 grids with dense clutter, probabilistic action slip, and capability-shaped costs. In both regimes, the agent first trains across multiple source environments to build the prior, then adapts to a novel user–environment combination over a limited number of episodes. In the Hard regime, the pretrained agent reaches the goal in 83.3% of test maps compared to 8.3% for a scratch agent, improves mean final return from approximately -499.9 to -125.7, and increases path efficiency from 0.083 to 0.583. In the Easy regime, pretrained and scratch agents achieve similar final performance, indicating that the prior does not hinder adaptation in simple settings. The method runs with low computational requirements, relying on tabular updates and short adaptation windows, and yields consistent benefits in challenging, sparse-reward environments, supporting rapid personalization for assistive navigation.

Keywords: Meta-learning, Assistive Navigation, Reinforcement Learning, Personalization

1 Introduction

Assistive navigation technologies, such as powered wheelchairs and mobility aids, must adapt to individual users [1, 2]. Users bring different motor capabilities and interaction patterns [3, 4]. Environments also change. Long training cycles are not practical in daily use [5].

We aim for fast adaptation in assistive navigation with meta-learning. We learn a prior across many tasks. The prior captures regularities in maps and in capability effects [6, 7]. The learner then adapts on a new task with a short budget. The system runs on standard hardware.

We treat user capability inside the reward. We use penalties for turning and backtracking. We also add action slip to model motor noise [8, 9]. This makes the learned strategies capability aware. The design stays simple and interpretable.

We test on a grid proxy that keeps compute low and still captures key problems in mobility [10, 11]. The proxy has obstacles, start and goal, action slip, and capability costs.

Our evaluation shows clear gains on Hard cases. Success rate rises from 8.3% to 83.3%. Final return and efficiency improve by large margins. Easy cases show parity at the end of training. The prior does not reduce performance when the task is simple.

Prior work on assistive navigation rarely bakes user capability into the learning objective and often needs long adaptation. We hypothesize that a capability-aware prior, learned through tabular Q-learning over diverse tasks, will cut adaptation time and raise success on difficult maps. Our Hard benchmark confirms this with 83.3% success for the pretrained agent versus 8.3% for scratch while Easy tasks remain at parity.

2 Related Work

2.1 Assistive Navigation Systems

Assistive navigation has moved from basic obstacle avoidance to shared control [1, 5]. Systems now aim to preserve user agency while helping when needed [11, 12]. Personalization is key, since users differ in motor control and interaction load [2, 13, 3, 10]. These trends motivate fast adaptation in practice.

2.2 Meta-Learning and Few-Shot Adaptation

Meta-learning improves adaptation to new tasks [14, 15]. Gradient-based methods learn initializations that adapt quickly [6, 7, 16]. In reinforcement learning, meta-learning speeds policy improvement using structure across tasks [17, 18]. Memory-based methods also help with quick updates [19, 17]. We study a tabular version built for capability-aware navigation. We keep the learner simple to support low compute. We also rely on standard Q-learning foundations [20, 21].

2.3 Capability-Aware Learning

Human-robot interaction benefits from models that reflect human limits and preferences [8, 22]. Capability-aware design improves shared autonomy and trust [23, 24]. In assistive settings this includes motor limits and cognitive load [9, 10]. Inverse reinforcement learning can reveal preferences but needs more data than we can afford during quick adaptation [25, 26]. We avoid that cost by learning a reusable prior that adapts with short runs.

3 Methodology

3.1 Problem Formulation

We model assistive navigation as a capability-conditioned MDP. The state is a grid cell $s = (r, c)$ with fixed start and goal per map. The agent can move up, down, left, or right. We inject action slip with probability p_{slip}. The slip replaces the intended action with a random action. This models control noise seen in real devices [12, 11].

We write the reward as

$$r(s, a, s') = r_{\text{env}}(s, a, s') - \lambda_{\text{turn}} \cdot \mathbf{1}_{\text{turn}} - \lambda_{\text{back}} \cdot \mathbf{1}_{\text{back}}. \tag{1}$$

The capability profile is $c = (\lambda_{\text{turn}}, \lambda_{\text{back}}, p_{\text{slip}})$. The penalties express the extra effort that some users face.

3.2 Environment Generation and Solvability

We sample random grids with obstacles. The Easy regime uses 8×8 with obstacle probability 0.20. The Hard regime uses 12×12 with obstacle probability 0.30. We enable action slip in Hard. We also enable capability costs in Hard. We filter out unsolvable maps. We run Breadth-First Search on the occupancy grid [27]. Only maps with a free path remain in the set.

3.3 Meta-Learning Algorithm

We use tabular Q-learning with a capability-aware prior. The base update is

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]. \tag{2}$$

We set $\alpha = 0.10$. We set $\gamma = 0.99$. We use ϵ-greedy exploration with $\epsilon_0 = 1.0$ and geometric decay 0.99 per episode [21, 20].

We learn the prior by averaging Q-values across short runs on many tasks. For the Easy prior we use 10 tasks with 100 episodes each. We use 8×8 grids with obstacle probability 0.25. For the Hard prior we use 20 tasks with 150 episodes each. We use 12×12 grids with obstacle probability 0.30. We randomize $\lambda_{\text{turn}} \in [0.5, 2.0]$, $\lambda_{\text{back}} \in [0.5, 2.0]$, and $p_{\text{slip}} \in [0.05, 0.20]$. The prior Q_{prior} encodes common navigation patterns that work across users.

3.4 Adaptation Protocol

We adapt the agent on a new map with either a random start or the learned prior. For Easy we run 200 episodes without capability costs. For Hard we run 60 episodes with a fixed profile $c^* = (1.5, 1.2, 0.15)$. After adaptation we run a greedy rollout with a 300-step cap. This setup matches a quick, practical tuning step before use.

4 Experimental Setup

4.1 Evaluation Metrics

We report five metrics. Reach is a binary success flag. Final Return is the cumulative reward on the rollout. Path Efficiency is the ratio of the shortest path length to the executed path length. SPL combines success with normalized path length [28]. Steps on Success measures the number of steps when the agent reaches the goal. These metrics reflect both success and efficiency.

4.2 Statistical Analysis

We estimate 95% confidence intervals with bootstrap resampling over environments. We use 2,000 resamples [29]. We test if the pretrained approach beats scratch with a one-sided paired t-test [30]. We also repeat experiments across seeds to check stability.

5 Results

5.1 Performance on Hard Navigation Tasks

Figure 1 shows the learning curves on Hard maps. The pretrained mean stays above scratch for all 60 episodes. The confidence bands separate over most of training.

Table 1 quantifies the gains. Success rises to 0.833 from 0.083 ($p = 2.4\text{e-}07$). Final Return improves to -125.729 from -499.925 ($p = 3.18\text{e-}07$). Path Efficiency rises to 0.583 from 0.083 ($p = 1.23\text{e-}07$). Steps on Success drops from 131.000 to 45.450.

Table 1: Hard benchmark results (12×12, obstacle prob 0.30, action slip, capability-shaped rewards). Mean ± 95% CI over 24 environments; paired one-sided t-test.

Metric	Scratch	Pretrained	p_{better}
Reach	0.083 ± 0.104	0.833 ± 0.146	2.4e-07
Final Return	-499.925 ± 49.575	-125.729 ± 71.069	3.18e-07
Path Efficiency	0.083 ± 0.011	0.583 ± 0.125	1.23e-07
Steps on Success	131.000 ± 14.000	45.450 ± 18.352	-

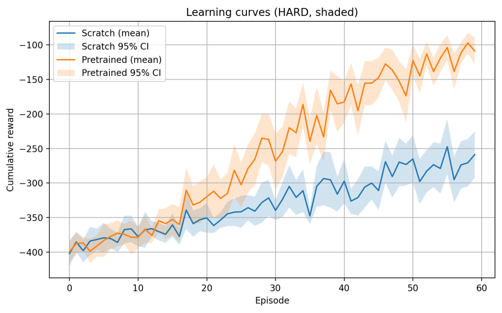

Figure 1: Learning curves for Hard navigation tasks. Pretrained (orange) consistently outperforms learning from scratch (blue) across 60 adaptation episodes.

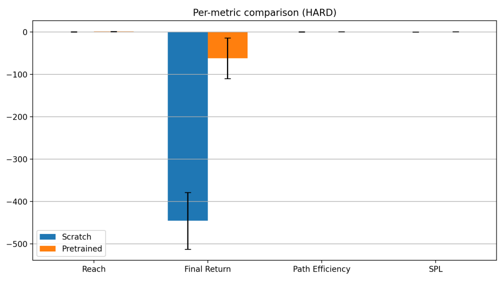

Figure 2: Performance comparison on Hard tasks across all metrics. Pretrained approach outperforms learning from scratch on all measures.

5.2 Qualitative Path Analysis

Figure 3 shows one Hard map. The scratch agent fails to reach the goal and loops in clutter. The pretrained agent reaches the goal along a direct route. The plot illustrates the practical benefit of the prior.

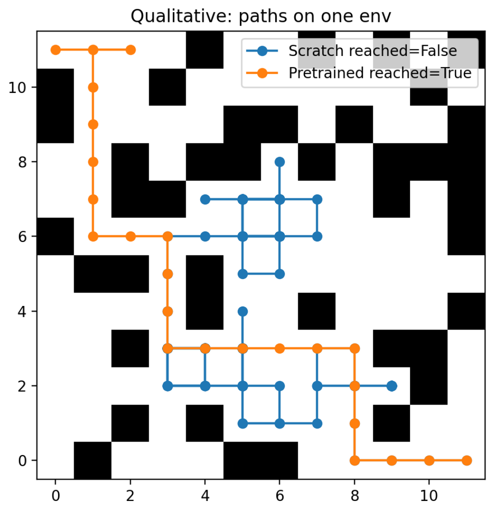

Figure 3: Example navigation paths on a Hard environment. Blue: learning from scratch (fails to reach goal). Orange: pretrained approach (successful, efficient path).

5.3 Performance on Easy Navigation Tasks

Figure 4 shows Easy curves. The pretrained mean starts higher. The gap closes with more episodes. Both approaches match near the end.

Table 2 reports parity. Success is 1.000 for both. Final Return and Path Efficiency also match. Steps on Success is the same.

5.4 Ablation Studies

We study two factors on Hard maps. First, the number of meta-training tasks. Second, the adaptation budget.

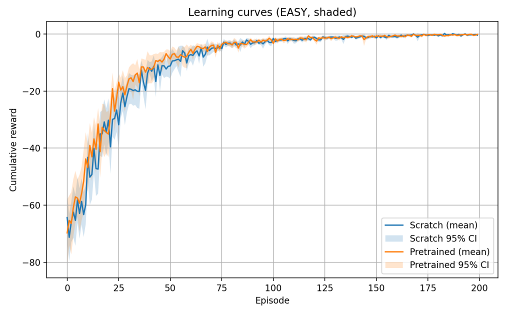

Figure 4: Learning curves for Easy navigation tasks. Both approaches converge to similar final performance.

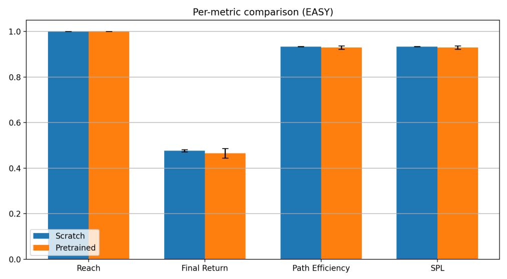

Figure 5: Performance comparison on Easy tasks. Both approaches achieve statistically equivalent performance.

Table 2: Easy benchmark results (8×8, obstacle prob 0.20). Mean ± 95% CI over 24 environments; paired one-sided t-test.

Metric	Scratch	Pretrained	p_{better}
Reach	1.000 ± 0.000	1.000 ± 0.000	nan
Final Return	0.465 ± 0.021	0.465 ± 0.021	nan
Path Efficiency	0.934 ± 0.001	0.934 ± 0.001	nan
Steps on Success	15.167 ± 0.208	15.167 ± 0.208	nan

Task count shows an inverted U shape. Moderate diversity gives the strongest prior. Very high diversity makes the prior too broad. The adaptation budget plot shows that most of the benefit appears by 60 episodes. This supports short, practical tuning in real use.

5.5 Reproducibility Analysis

We repeat the study with multiple seeds and a fixed test set. Tables 3 and 4 show the results. Easy remains at parity. Hard shows large gains with strong significance.

Table 3: Easy benchmark across 3 seeds with fixed test set. Mean ± 95% CI; one-sided p_{better}.

Metric	Scratch	Pretrained	p_{better}
Reach	1.000 ± 0.000	1.000 ± 0.000	nan
Final Return	0.465 ± 0.023	0.465 ± 0.023	nan
Path Efficiency	0.934 ± 0.001	0.934 ± 0.001	nan
SPL (higher better)	0.934 ± 0.001	0.934 ± 0.001	nan

6 Discussion

6.1 Implications for Assistive Technology

Fast personalization matters in assistive devices. The prior lifts success rates on difficult maps without hurting simple ones. The method is also practical. It runs with tabular updates and short adaptation. It fits laptop-level compute [2, 1]. The capability parameters give a handle for interpretation [9, 10]. Clinicians can reason about λ_{turn} and λ_{back} and adjust training if needed.

6.2 Limitations and Future Work

The grid proxy is simple. Real navigation involves continuous control and moving obstacles. Sensors add noise and latency [5, 11]. We plan to test in richer simulators next. We also plan to extend

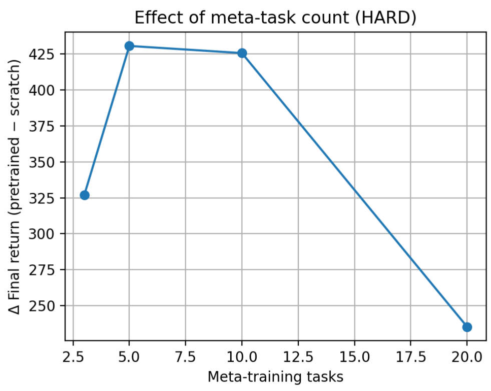

Figure 6: Ablation of meta-training task count. Gains rise from 3 to 5 tasks, stay high at 10, then decline at 20.

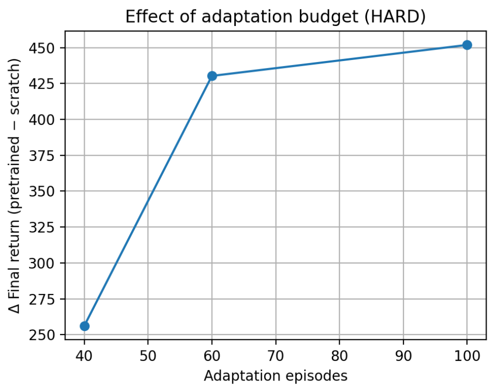

Figure 7: Ablation of adaptation budget. The largest jump occurs from 40 to 60 episodes. The gain from 60 to 100 is smaller.

Table 4: Hard benchmark across 3 seeds with fixed test set. Mean $\pm$ 95% CI; one-sided p_{better}.

Metric	Scratch	Pretrained	p_{better}
Reach	0.125 ± 0.076	0.847 ± 0.104	2.2e-12
Final Return	-472.710 ± 25.919	-109.657 ± 56.748	4.24e-12
Path Efficiency	0.089 ± 0.011	0.646 ± 0.108	5.21e-10
SPL (higher better)	0.023 ± 0.016	0.634 ± 0.119	2.51e-10

to multiple users per device with quick profile switching. The ablation results suggest that task selection matters. A curriculum for meta-training may help [6, 7].

6.3 Broader Impact

Rapid personalization can improve quality of life for users who rely on mobility aids [4, 3]. Learning systems must also be safe and reliable. The capability-aware design improves transparency, yet we still need strong safeguards and fail-safe behavior [13].

7 Conclusion

We presented a capability-aware meta-learning method for assistive navigation. The prior helps the agent adapt fast to new users and new maps. On Hard tasks the pretrained agent reaches 83.3% success while scratch reaches 8.3%. Easy tasks show parity. The method is simple and efficient. It uses tabular Q-learning and short adaptation. The results support practical personalization with clear benefits and low cost.

References

[1] R. C. Simpson. Smart wheelchairs: A literature review. *Journal of Rehabilitation Research & Development*, 42(4):423–436, 2005.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, 2018.

[3] R. A. Cooper, H. Ohnabe, and D. A. Hobson. Rehabilitation engineering and assistive technology: trajectories, technologies, and tools. *Journal of Rehabilitation Research & Development*, 47(4):vii–xiv, 2010.

[4] L. Fehr, W. E. Langbein, and S. B. Skaar. Adequacy of power wheelchair control interfaces for persons with severe disabilities: a clinical survey. *Journal of Rehabilitation Research & Development*, 37(3):353–360, 2000.

[5] E. Demeester, A. Huntemann, D. Vanhooydonck, G. Vanacker, H. Van Brussel, and M. Nuttin. User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving. *Autonomous Robots*, 24(2):193–211, 2008.

[6] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

[7] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. In *arXiv preprint arXiv:1803.02999*, 2018.

[8] A. D. Dragan and S. S. Srinivasa. Policy blending and shared control for assistive robotics. *IEEE Transactions on Robotics*, 29(4):803–820, 2013.

[9] S. Jain and B. Argall. Assistive robotic manipulation through shared autonomy and a body-machine interface. *IEEE International Conference on Robotics and Automation*, pages 2633–2640, 2013.

[10] Z. Wang, K. Mulling, M. P. Deisenroth, H. Ben Amor, D. Vogt, B. Scholkopf, and J. Peters. Assistive robotics for independent living: Recent progress and open challenges. *Robotics and Autonomous Systems*, 101:48–60, 2017.

[11] T. Carlson and J. d. R. Millán. Brain-controlled wheelchairs: a robotic architecture. *IEEE Robotics & Automation Magazine*, 20(1):65–73, 2012.

[12] J.-H. Kim, J.-H. Lim, K.-H. Lee, H.-S. Ro, and H.-M. Rhee. Comparison of p300-based brain-computer interfaces for wheelchair control. In *International Conference on Control, Automation and Systems*, pages 2130–2135, 2012.

[13] S. Reddy, A. D. Dragan, and S. Levine. Shared autonomy via deep reinforcement learning. In *Robotics: Science and Systems*, 2018.

[14] J. Schmidhuber. Evolutionary principles in self-referential learning. *Diploma thesis, Institut für Informatik, Technische Universität München*, 1987.

[15] S. Thrun and L. Pratt. Learning to learn: Introduction and overview. *Learning to learn*, pages 3–17, 1998.

[16] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[17] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. In *arXiv preprint arXiv:1611.02779*, 2016.

[18] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. In *arXiv preprint arXiv:1611.05763*, 2016.

[19] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850, 2016.

[20] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[21] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *MIT press*, 2018.

[22] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for autonomous cars that leverage effects on human actions. *Robotics: Science and Systems*, 2016.

[23] S. Nikolaidis, R. Ramakrishnan, K. Gu, and A. Dragan. Human-robot mutual adaptation in shared autonomy. *ACM/IEEE International Conference on Human-Robot Interaction*, pages 294–302, 2017.

[24] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, and S. Srinivasa. Planning with trust for human-robot collaboration. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 307–315, 2018.

[25] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, pages 1–8, 2004.

[26] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *International Joint Conference on Artificial Intelligence*, pages 2586–2591, 2007.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3 edition, 2009.

[28] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. In *arXiv preprint arXiv:1807.06757*, 2018.

[29] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1994.

[30] Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.