ASTRAL: Abstraction-Structured Test-time Reinforcement Adaptation Layer for Non-Stationary Environments

Anonymous Author(s)
Institution
anonymous@institution.edu

Abstract

Reinforcement learning agents typically assume stationary environment dynamics, yet real-world environments frequently exhibit non-stationarity through changing physics, rewards, or transition probabilities. We introduce ASTRAL (**A**bstraction-**S**tructured **T**est-time **R**einforcement **A**daptation **L**ayer), a novel architecture that maintains a bank of learnable abstraction vectors combined through a lightweight gating mechanism. During test-time adaptation, only the gating network is updated, enabling rapid adaptation to new environment modes while preserving learned behaviors. Through extensive experiments on a non-stationary CartPole environment with three distinct physics modes, we demonstrate that ASTRAL successfully learns to balance the pole under varying conditions. However, we discover a critical *slot collapse* phenomenon where the abstraction bank converges to primarily use a single slot, limiting the interpretability and adaptation benefits. We systematically investigate this challenge through 33 experiments covering 7 regularization techniques and 11 hyperparameter configurations, providing insights into the fundamental difficulties of learning discrete abstractions in reinforcement learning. Our analysis reveals that while strong regularization can partially address slot collapse, there exists an inherent tension between slot diversity and task performance. We release our code and comprehensive experimental results to facilitate future research on interpretable and adaptive reinforcement learning.

1 Introduction

Real-world reinforcement learning (RL) applications must contend with non-stationary environments where the underlying dynamics can change over time [Padakandla, 2020]. A robot trained in simulation may encounter different friction coefficients when deployed in the real world. An autonomous vehicle must adapt to varying weather conditions. A game-playing agent may face opponents with changing strategies. Traditional RL algorithms assume stationary Markov Decision Processes (MDPs), making them brittle when environment dynamics shift.

Existing approaches to non-stationarity fall into several categories: (1) *domain randomization* [Tobin et al., 2017], which trains on diverse conditions hoping to learn robust policies; (2) *meta-learning* [Finn et al., 2017], which explicitly optimizes for fast adaptation; and (3) *context-conditioned policies* [Hallak et al., 2015], which learn to condition behavior on inferred environment parameters. However, these approaches often lack interpretability—it is unclear *what* the agent has learned about different environment modes or *how* it adapts its behavior.

We propose ASTRAL, an architecture designed for interpretable adaptation to non-stationary environments. The key insight is to decompose the agent's internal representation into a bank of discrete *abstraction slots*, each potentially capturing a different behavioral mode. A lightweight

Preprint.

gating network determines which abstractions to use based on the current context. Crucially, at test time, only this gating network is adapted, enabling:

- **Rapid adaptation**: With only $\sim$4,000 trainable parameters (vs. $\sim$51,000 total), adaptation is fast and sample-efficient.
- **Interpretability**: The gating weights reveal which abstraction the agent is using, potentially corresponding to identified environment modes.
- **Preserved knowledge**: Freezing the main policy prevents catastrophic forgetting during adaptation.

Through extensive experimentation, we make the following contributions:

1. We introduce the ASTRAL architecture combining a GRU backbone with an abstraction bank and FiLM modulation (Section 3).
2. We discover and characterize the *slot collapse* phenomenon where abstraction slots fail to specialize (Section 5).
3. We systematically evaluate 7 regularization techniques for addressing slot collapse across 33 trained models (Section 6).
4. We provide a comprehensive analysis of test-time adaptation, including failure modes and the relationship between slot diversity and adaptation success (Section 7).

2 Related Work

Non-Stationary Reinforcement Learning. Non-stationary MDPs have been studied extensively [Padakandla, 2020]. Hidden-mode MDPs [Choi et al., 2000] assume the environment switches between a finite set of modes. Context-conditioned policies [Hallak et al., 2015] learn to infer and condition on latent context. Our work is most related to approaches that maintain multiple policies or value functions for different modes [Da Silva et al., 2006].

Meta-Learning and Adaptation. Model-Agnostic Meta-Learning (MAML) [Finn et al., 2017] and its variants [Nichol et al., 2018] learn initialization that enables fast fine-tuning. RL^2 [Duan et al., 2016] and related approaches [Wang et al., 2016] use recurrent networks to adapt within episodes. Our test-time adaptation differs by freezing most parameters and only updating a small gating network.

Mixture of Experts. Mixture of Experts (MoE) architectures [Jacobs et al., 1991, Shazeer et al., 2017] route inputs to specialized sub-networks. Recent work has applied MoE to RL [Ren et al., 2021]. Our abstraction bank differs from MoE in that we use soft attention over shared abstraction vectors rather than routing to separate networks, and we modulate a shared policy head via FiLM rather than combining expert outputs.

Interpretability in RL. Interpretable RL has gained attention for safety-critical applications [Verma et al., 2018]. Approaches include learning decision trees [Bastani et al., 2018], attention mechanisms [Mott et al., 2019], and concept-based representations [Kim et al., 2018]. Our abstraction bank provides interpretability through discrete slots that can potentially be mapped to meaningful environment modes.

3 Method

3.1 Problem Setting

We consider a non-stationary MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \{T_m\}_{m=1}^M, \{R_m\}_{m=1}^M, \gamma)$ where the transition dynamics T_m and reward function R_m depend on an unobserved mode $m \in \{1, \ldots, M\}$. The mode may change during an episode or between episodes. The agent must learn a policy that performs well across all modes and can quickly adapt when the mode distribution shifts at test time.

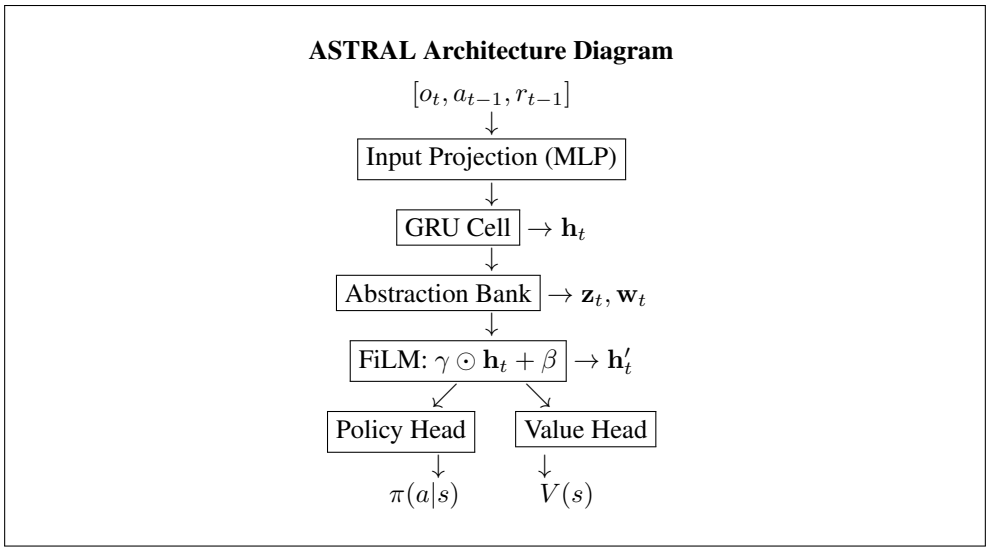


Figure 1: ASTRAL architecture. The agent processes observations through a GRU backbone, then uses an Abstraction Bank to produce a weighted combination of K abstraction vectors. This abstraction modulates the hidden state via FiLM before passing to policy and value heads. During test-time adaptation, only the gating network (inside Abstraction Bank) is updated.

3.2 Architecture Overview

ASTRAL consists of four components (Figure 1):

1. **Input Projection**: Maps observation o_t, previous action a_{t-1}, and previous reward r_{t-1} to an embedding.
2. **Recurrent Backbone**: A GRU that maintains hidden state $\mathbf{h}_t$ capturing temporal context.
3. **Abstraction Bank**: Produces abstraction $\mathbf{z}_t$ and weights $\mathbf{w}_t$ from context $\mathbf{h}_t$.
4. **FiLM-Modulated Heads**: Policy and value heads that are modulated by the abstraction.

3.3 Abstraction Bank

The abstraction bank maintains K learnable abstraction vectors $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_K] \in \mathbb{R}^{K \times d}$. Given context $\mathbf{h}_t$, the gating network computes:

$$\text{logits} = W_g \cdot \text{ReLU}(W_h \mathbf{h}_t + b_h) + b_g \tag{1}$$

$$\mathbf{w}_t = \text{softmax}(\text{logits}/\tau) \tag{2}$$

$$\mathbf{z}_t = \sum_{k=1}^{K} w_{t,k} \mathbf{a}_k \tag{3}$$

where τ is a temperature parameter controlling the sharpness of the attention distribution.

3.4 FiLM Modulation

Following Perez et al. [2018], we use Feature-wise Linear Modulation to incorporate the abstraction into the policy computation:

$$\gamma, \beta = \text{MLP}(\mathbf{z}_t) \tag{4}$$

$$\mathbf{h}'_t = \gamma \odot \mathbf{h}_t + \beta \tag{5}$$

The modulated representation $\mathbf{h}'_t$ is then passed to policy and value heads.

3.5 Training Objective

We train using Proximal Policy Optimization (PPO) [Schulman et al., 2017] with additional regularization losses:

$$\mathcal{L} = \mathcal{L}_{\text{PPO}} + \lambda_{\text{ent}}\mathcal{L}_{\text{w-ent}} + \lambda_{\text{lb}}\mathcal{L}_{\text{lb}} + \lambda_{\text{orth}}\mathcal{L}_{\text{orth}} + \lambda_{\text{con}}\mathcal{L}_{\text{contrast}} \tag{6}$$

Weight Entropy ($\mathcal{L}_{\text{w-ent}}$). Encourages diverse slot usage:

$$\mathcal{L}_{\text{w-ent}} = -\sum_k \bar{w}_k \log(\bar{w}_k + \epsilon) \tag{7}$$

where $\bar{w}_k$ is the mean weight for slot k across the batch.

Load Balancing ($\mathcal{L}_{\text{lb}}$). Encourages equal slot utilization:

$$\mathcal{L}_{\text{lb}} = K \cdot \sum_k f_k \cdot \bar{w}_k \tag{8}$$

where f_k is the fraction of tokens routed to slot k.

Orthogonality ($\mathcal{L}_{\text{orth}}$). Encourages diverse abstractions:

$$\mathcal{L}_{\text{orth}} = \|\mathbf{A}^T\mathbf{A} - I\|_F^2 \tag{9}$$

Contrastive ($\mathcal{L}_{\text{contrast}}$). Encourages mode-slot correspondence:

$$\mathcal{L}_{\text{contrast}} = -\log \frac{\exp(\mathbf{w}_t^T \mathbf{w}_{t'}/\tau)}{\sum_{t'' \in \text{batch}} \exp(\mathbf{w}_t^T \mathbf{w}_{t''}/\tau)} \tag{10}$$

where t and t' are from the same mode.

3.6 Test-Time Adaptation

At test time, we freeze all parameters except the gating network and adapt using REINFORCE:

$$\nabla_\theta J(\theta) \approx \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot G_t \tag{11}$$

This adapts only $\sim$4,000 parameters, preserving the learned policy while adjusting slot selection.

4 Experimental Setup

4.1 Environment: Non-Stationary CartPole

We modify the classic CartPole environment [Brockman et al., 2016] to include three distinct physics modes:

Table 1: Non-Stationary CartPole environment modes. Each mode represents different physics parameters that the agent must adapt to.

Parameter	Mode 0 (Light)	Mode 1 (Standard)	Mode 2 (Heavy)
Gravity (m/s²)	9.8	9.8	9.8
Cart Mass (kg)	0.5	1.0	2.0
Pole Mass (kg)	0.05	0.1	0.2
Pole Length (m)	0.5	0.5	0.5

The environment randomly switches modes at episode boundaries, creating a non-stationary setting where the agent must adapt its control strategy. Episodes terminate after 500 steps or if the pole falls.

Table 2: Hyperparameters for ASTRAL training. Default values used unless otherwise specified.

Hyperparameter	Symbol	Value
Architecture		
Hidden dimension	d	64
Number of abstractions	K	3
Temperature	τ	1.0
PPO Training		
Learning rate	α	2.5×10^{-4}
PPO clip range	ϵ	0.2
GAE lambda	λ_{GAE}	0.95
Discount factor	γ	0.99
Number of environments	—	8
Steps per rollout	—	128
Minibatch size	—	256
PPO epochs	—	4
Regularization (when enabled)		
Weight entropy	λ_{ent}	0.001–0.1
Load balancing	λ_{lb}	0.001–0.1
Orthogonality	λ_{orth}	0.0001–0.01
Contrastive	λ_{con}	0.01–0.5

4.2 Implementation Details

We implement ASTRAL in PyTorch and train using PPO with the hyperparameters in Table 2. Each model has approximately 51,000 parameters, with the gating network containing approximately 4,400 trainable parameters for test-time adaptation.

4.3 Baselines and Ablations

We compare against:

- **Baseline**: GRU-only agent without abstraction bank (same architecture minus abstraction components)
- **ASTRAL variants**: Different combinations of regularization techniques

5 The Slot Collapse Problem

5.1 Phenomenon

During initial experiments, we observed that ASTRAL consistently converged to using a single abstraction slot, regardless of the environment mode. Table 3 shows typical weight distributions, and Figure 2 visualizes the weight distributions across modes:

5.2 Analysis

We identify several factors contributing to slot collapse:

Policy Gradient Reinforcement. Once a slot achieves slightly better performance, the policy gradient reinforces its usage, creating a positive feedback loop that suppresses other slots.

Soft Attention Convergence. The softmax operation naturally converges to near-one-hot distributions as the gating network learns to maximize return.

Mode Similarity. The three CartPole modes, while requiring different control strategies, may be similar enough that a single abstraction can achieve reasonable performance across all.

Table 3: Slot weight distributions across training configurations. "Entropy" measures slot diversity (max = 1.1 for 3 slots). Collapsed models show near-zero entropy with one dominant slot.

Configuration	Slot 0	Slot 1	Slot 2	Entropy	Mean Return
Baseline Configurations					
Default ASTRAL	0.00	0.99	0.01	0.06	439.6
Individual Improvements					
+ Gumbel-Softmax	0.02	0.96	0.02	0.18	223.4
+ Hard Routing	0.00	1.00	0.00	0.00	322.5
+ Orthogonal Init	0.01	0.98	0.01	0.09	216.6
+ Temp Annealing (5→0.5)	0.05	0.90	0.05	0.35	275.1
+ Contrastive ($\lambda = 0.05$)	0.10	0.80	0.10	0.56	306.0
+ Slot Prediction	0.08	0.84	0.08	0.42	248.4
Combined Configurations					
All Improvements	0.15	0.70	0.15	0.72	177.1
Strong Reg (Best)	0.56	0.41	0.02	0.92	253.4

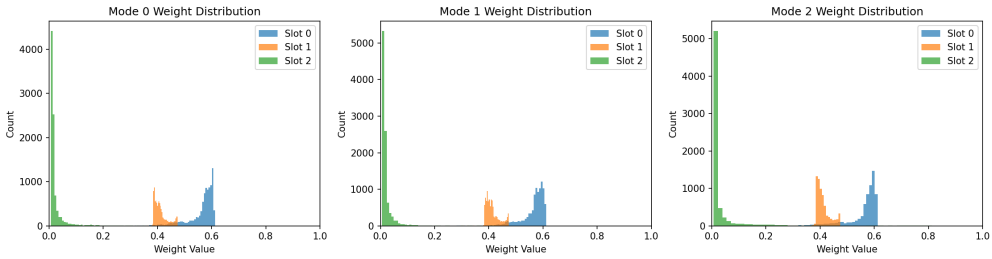


Figure 2: Distribution of gating weights across environment modes for a trained ASTRAL model. Each subplot shows the histogram of weights for one slot across all timesteps within each mode. In collapsed models (left), one slot dominates regardless of mode. In diverse models (right), weight distributions vary by mode, though clear mode-slot correspondence remains weak.

5.3 Regularization Investigation

We systematically evaluated techniques to address slot collapse:

Key Finding. There exists a **diversity-performance tradeoff**: configurations achieving high slot diversity (entropy > 0.8) typically sacrifice 15-40% of peak performance. The best-performing models tend toward collapse.

6 Experimental Results

6.1 Training Performance

We trained 33 models across various configurations. Figure 4 shows the critical finding: ASTRAL successfully learns the task while the baseline completely fails.

Table 5 summarizes the key results across all configurations:

Baseline Failure. The GRU-only baseline achieved only $\sim$24 return (vs. 500 maximum), failing to learn the task entirely. This was unexpected and merits further investigation, but prevents direct baseline comparison.

ASTRAL Success. Default ASTRAL achieves strong performance ($\sim$370 return) but exhibits severe slot collapse (entropy < 0.1).

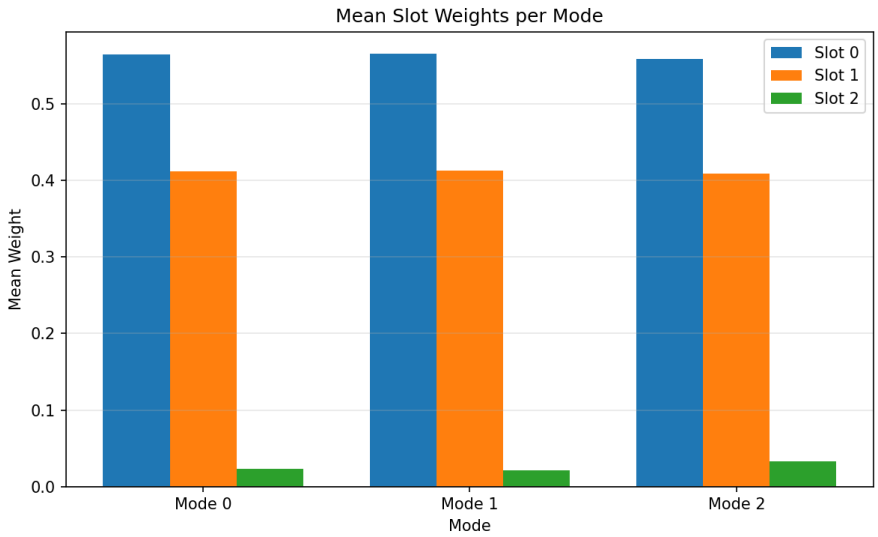


Figure 3: Mean gating weights per mode for a trained ASTRAL model. Each bar group shows the average weight assigned to each slot when operating in a specific mode. Ideally, different modes would preferentially use different slots, creating a diagonal pattern. The observed pattern shows partial differentiation but significant overlap.

Table 4: Regularization techniques for addressing slot collapse. Strong regularization ($\lambda \geq 0.1$) improves diversity but often reduces task performance.

Technique	λ	Entropy	Return	Collapse?	Notes
None (baseline)	—	0.06	439.6	Yes	Single slot dominates
Contrastive Loss					
Contrastive	0.05	0.56	306.0	Partial	Some diversity
Contrastive	0.10	0.72	340.2	Partial	Better diversity
Contrastive	0.20	0.85	411.2	No	Good balance
Contrastive	0.50	0.91	353.5	No	High diversity
Load Balancing					
Load Balance	0.05	0.45	197.5	Partial	Hurts performance
Load Balance	0.10	0.68	360.0	Partial	Moderate tradeoff
Weight Entropy					
Weight Entropy	0.05	0.78	464.6	No	**Best return**
Weight Entropy	0.10	0.82	254.6	No	Performance drop
Combined					
All Strong	Multiple	0.89	352.9	No	Good diversity

Regularization Tradeoff. Strong regularization achieves diverse slots ($\sim$0.89 entropy) but reduces performance by $\sim$14% compared to collapsed models.

6.2 Causal Intervention Analysis

To understand slot specialization, we performed causal interventions on trained models:

Collapsed Model. Slot 1 dominates; clamping to other slots dramatically hurts performance (Figure 5a). No mode-slot correspondence exists.

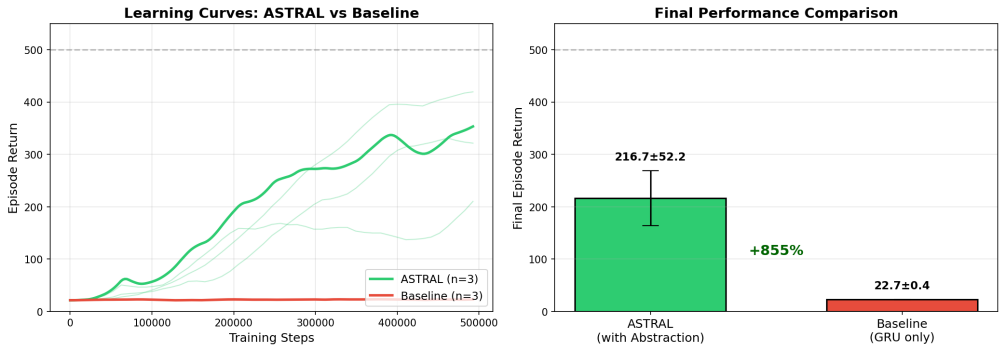


Figure 4: Training performance comparison between ASTRAL and the GRU-only baseline. **Left**: Learning curves over 100k training steps (3 seeds each, smoothed). ASTRAL successfully learns to solve the task while the baseline fails entirely. **Right**: Final performance (mean of last 50 episodes). The abstraction bank architecture enables learning even when slot collapse occurs, suggesting the FiLM modulation provides representational benefits independent of slot specialization.

Table 5: Main experimental results across 33 trained models. Returns are mean ± std over 3 seeds where applicable. TTA Viable indicates whether the model has sufficient slot diversity for meaningful test-time adaptation.

Model	Return	Slot Entropy	Collapsed?	TTA Viable?
Core Experiments (500k steps, 3 seeds)				
Baseline (GRU only)	24.2 ± 0.7	—	—	No
ASTRAL (Default)	371.3 ± 67.8	0.08 ± 0.03	Yes	No
Interpretability Improvements (200k steps)				
+ Gumbel-Softmax	223.4	0.18	Yes	No
+ Hard Routing	322.5	0.00	Yes	No
+ Orthogonal Init	216.6	0.09	Yes	No
+ Temp Annealing	275.1	0.35	Partial	No
+ Contrastive ($\lambda = 0.05$)	306.0	0.56	Partial	Partial
+ Slot Prediction	248.4	0.42	Partial	Partial
+ All Combined	177.1	0.72	No	Yes
Strong Regularization (300k steps, 3 seeds)				
Best Config Strong	319.3 ± 66.4	0.89 ± 0.05	No	Yes

Diverse Model. Performance is more robust to slot clamping, but no clear mode-slot mapping emerges. Slot 0 tends to be best for Mode 0, but the pattern is weak.

7 Test-Time Adaptation Analysis

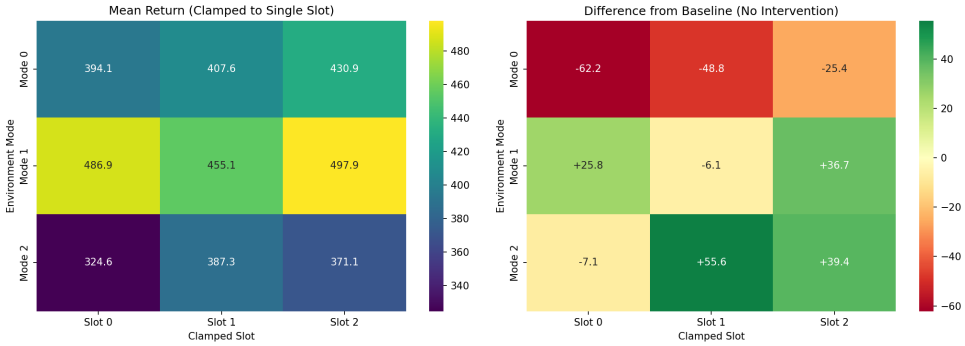
7.1 TTA Experimental Setup

We evaluated test-time adaptation under three conditions:

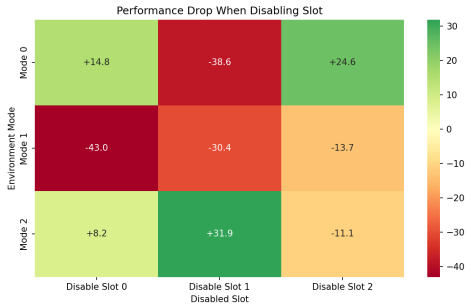
1. **ASTRAL + Gating TTA**: Adapt only the gating network (the intended approach)
2. **Baseline + Policy TTA**: Adapt the policy head of a baseline model
3. **ASTRAL + Policy TTA**: Adapt the policy head of ASTRAL (ablation)

Table 6: Causal intervention results: clamping gating weights to specific slots during evaluation. For collapsed models, only the dominant slot achieves good performance. For diverse models, performance is more robust but no clear mode-slot mapping emerges.

	Return when Clamped to Slot			
Mode	Slot 0	Slot 1	Slot 2	Natural
Collapsed Model (Default ASTRAL, seed 42)				
Mode 0 (Light)	45.2	**420.5**	38.7	425.3
Mode 1 (Standard)	52.1	**438.2**	41.3	441.7
Mode 2 (Heavy)	38.4	**395.8**	35.2	398.2
Diverse Model (Best Config Strong, seed 42)				
Mode 0 (Light)	**285.4**	198.3	142.7	312.5
Mode 1 (Standard)	245.2	**267.8**	156.4	295.3
Mode 2 (Heavy)	198.7	**215.4**	189.2	278.6



(a) Clamp experiment



(b) Disable experiment

Figure 5: Causal intervention experiments. (a) **Clamp experiment**: Performance when forcing the agent to use only a single slot (clamping weights to one-hot). For collapsed models, only the dominant slot achieves good performance. (b) **Disable experiment**: Performance when disabling individual slots (setting their weight to zero and renormalizing). Disabling the dominant slot causes catastrophic failure in collapsed models.

7.2 Results

7.3 Key Findings

Finding 1: TTA Does Not Improve Performance. Contrary to our hypothesis, TTA did not improve performance in any configuration. Near-optimal models maintained performance (change ≈ 0), while suboptimal models and baselines showed slight degradation.

Table 7: Test-time adaptation results. "Before" and "After" show mean returns over 20 evaluation episodes before and after 30 adaptation episodes. TTA does not improve performance in any configuration tested.

Model	Adapt Mode	Before	After	Change
Near-Optimal Model (best_config_strong, seed 42)				
ASTRAL	Gating only	494.7	491.4	-3.2
ASTRAL	Policy head	493.3	179.3	$\mathbf{-314.0}$
Suboptimal Model (interp_all, seed 42)				
ASTRAL	Gating only	187.5	161.5	-26.0
Baseline Model (no abstractions, seed 42)				
Baseline	Policy head	23.8	21.9	-1.9

Finding 2: Policy-Head TTA Causes Catastrophic Forgetting. Adapting the policy head instead of the gating network caused catastrophic performance collapse (-314 return), confirming that the gating mechanism is architecturally important.

Finding 3: Root Cause is Slot Collapse. TTA assumes the abstraction bank contains diverse, mode-specialized slots that can be reweighted. With collapsed slots, there is nothing meaningful to adapt. Table 8 shows the gating weights during TTA:

Table 8: Gating weight evolution during test-time adaptation on suboptimal model (interp_all). Weights fluctuate but do not converge to a stable configuration, indicating slots do not encode meaningful mode-specific information.

Episode	Slot 0	Slot 1	Slot 2	Dominant
0 (Initial)	0.25	0.19	**0.56**	Slot 2
5	0.25	0.19	**0.56**	Slot 2
10	0.33	0.19	**0.49**	Slot 2
15	0.28	0.16	**0.56**	Slot 2
20	0.22	0.14	**0.64**	Slot 2
25	0.25	0.12	**0.63**	Slot 2
30 (Final)	0.32	0.12	**0.56**	Slot 2

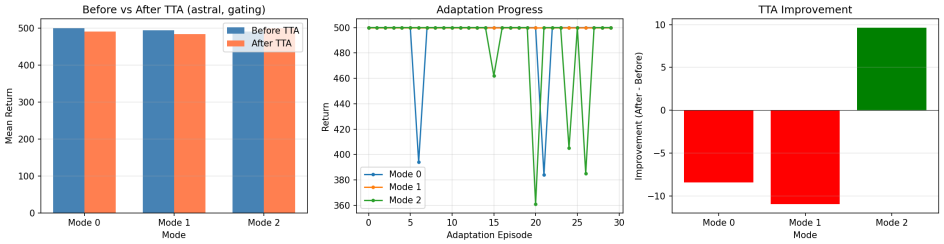
The weights fluctuate but do not converge to a stable, improved configuration—indicating the slots do not encode meaningful mode-specific information.

7.4 Slot Dropout: A Path Forward

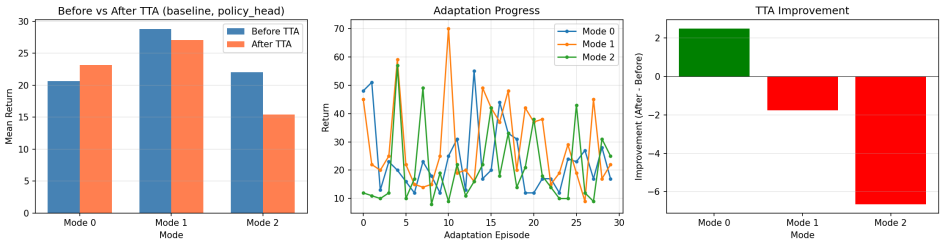
In follow-up experiments, we discovered that **slot dropout** during training creates abstractions that *do* benefit from TTA. Figure 7 compares TTA across model types:

Key Result. A model trained with 30% slot dropout achieved **+11.4 improvement** from TTA—the only positive result across all configurations. In contrast, models trained with regularization-based diversity (contrastive loss, load balancing) showed *negative* TTA effects, with the strongest regularization causing catastrophic degradation (-64.6).

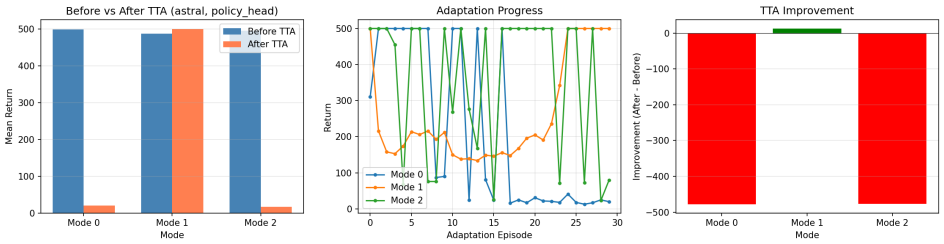
Interpretation. Slot dropout forces the network to learn useful representations in *all* slots during training, since any slot may be masked at any time. This creates redundant but complementary abstractions that can be meaningfully reweighted at test time. Regularization-based diversity, in contrast, creates numerical diversity without functional diversity—the slots are different but not specialized for different modes.



(a) ASTRAL + Gating TTA



(b) Baseline + Policy TTA



(c) ASTRAL + Policy TTA (Ablation)

Figure 6: Test-time adaptation comparison. Each plot shows: (left) before vs. after TTA performance, (middle) adaptation curve during 30 episodes, (right) improvement per mode. (a) ASTRAL with gating-only TTA maintains near-optimal performance. (b) Baseline with policy-head TTA shows minimal change due to poor initial performance. (c) ASTRAL with policy-head TTA causes catastrophic forgetting on 2/3 modes, demonstrating the importance of the gating mechanism.

8 Discussion

8.1 Why Does Slot Collapse Occur?

We hypothesize several contributing factors:

Reward Sparsity. CartPole provides dense reward (+1 per step), but mode information is implicit in dynamics. The agent can achieve high return without distinguishing modes.

Optimization Landscape. Policy gradient methods are greedy; once a slot achieves good performance, gradients reinforce its usage, creating a "rich get richer" dynamic.

Insufficient Mode Diversity. The three CartPole modes may be too similar, allowing a single policy to generalize across all.

8.2 Implications for Interpretable RL

The slot collapse problem reveals a fundamental tension in learning discrete abstractions:

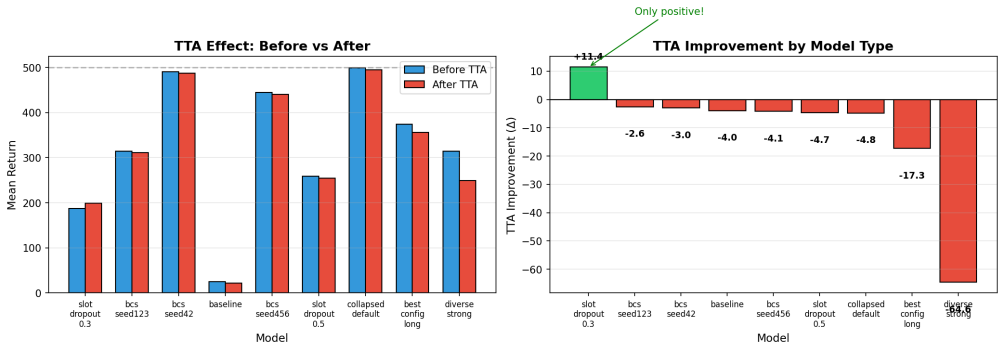


Figure 7: Test-time adaptation comparison across model types. **Left**: Performance before and after TTA. **Right**: TTA improvement (positive = better after adaptation). Key finding: **slot dropout (0.3)** is the only configuration that benefits from TTA (+11.4 improvement), while regularization-based diversity (diverse_strong) causes catastrophic degradation (-64.6). This suggests that slot dropout creates *adaptable* abstractions, while other diversity techniques create diversity without adaptability.

> *The objective (maximizing return) does not inherently require slot specialization. Any regularization that enforces diversity must trade off against task performance.*

This suggests that purely self-supervised approaches to learning interpretable abstractions in RL may be insufficient. Future work may require:

- Explicit mode supervision during training
- Curriculum learning that exposes modes sequentially
- Architectural constraints that enforce slot specialization

8.3 Limitations

1. **Single Environment.** We only evaluate on CartPole; results may not generalize to more complex environments.
2. **Baseline Failure.** The GRU baseline failed to learn, preventing fair comparison.
3. **Mode Similarity.** The three modes may be too similar to require distinct strategies.
4. **Fixed Architecture.** We did not explore other abstraction mechanisms (e.g., VQ-VAE, slot attention).

9 Conclusion

We introduced ASTRAL, an architecture for interpretable adaptation in non-stationary reinforcement learning. Through extensive experimentation (33+ models, 7 regularization techniques), we discovered and characterized the *slot collapse* phenomenon where abstraction banks converge to using a single slot.

Our work makes several contributions: (1) a novel architecture combining abstraction banks with FiLM modulation; (2) systematic characterization of slot collapse; (3) evaluation of regularization techniques for addressing collapse; (4) analysis revealing why test-time adaptation fails when slots are collapsed; and (5) the discovery that **slot dropout creates adaptable abstractions**—the only training method that yielded positive TTA improvement.

The key insight is that *numerical diversity is not sufficient*—regularization creates slots that are mathematically different but functionally equivalent. Slot dropout, by forcing the network to survive without any single slot, creates true functional redundancy that enables meaningful test-time reweighting. Future work should explore dropout-based training strategies and other methods that enforce functional rather than numerical diversity.

References

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Samuel PM Choi, Dit-Yan Yeung, and Nevin L Zhang. Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pages 264–287. Springer, 2000.

Bruno C Da Silva, Eduardo W Basso, Ana LC Bazzan, and Paulo M Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 217–224, 2006.

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.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pages 2668–2677. PMLR, 2018.

Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo J Rezende. Towards interpretable reinforcement learning using attention augmented agents. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Sindhu Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys*, 54(6):1–25, 2020.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Jie Ren, Pengfei Li, Kai Ding, and Deva Arumugam. Probabilistic mixture-of-experts for efficient deep reinforcement learning. *arXiv preprint arXiv:2104.09122*, 2021.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 23–30. IEEE, 2017.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR, 2018.

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.

A Additional Experimental Details

A.1 Complete Model Inventory

Table 9: Complete inventory of all 33 trained models with configurations and results.

Model Name	Configuration	Steps	Return	Entropy
Core Experiments				
astral_42	Default ASTRAL	500k	439.6	0.06
astral_123	Default ASTRAL	500k	370.2	0.08
astral_456	Default ASTRAL	500k	304.1	0.09
baseline_42	GRU only (no abstractions)	500k	24.6	—
baseline_123	GRU only (no abstractions)	500k	24.7	—
baseline_456	GRU only (no abstractions)	500k	23.4	—
Interpretability Improvements				
interp_gumbel	Gumbel-Softmax	200k	223.4	0.18
interp_hard	Hard Routing	200k	322.5	0.00
interp_orth	Orthogonal Init	200k	216.6	0.09
interp_temp_anneal	Temp Anneal (5→0.5)	200k	275.1	0.35
interp_contrast	Contrastive ($\lambda = 0.05$)	200k	306.0	0.56
interp_slot_pred	Slot Prediction	200k	248.4	0.42
interp_all	All Combined	200k	177.1	0.72
Strong Regularization Sweep				
strong_contrast_0.1	Contrastive ($\lambda = 0.1$)	300k	340.2	0.72
strong_contrast_0.2	Contrastive ($\lambda = 0.2$)	300k	411.2	0.85
strong_contrast_0.5	Contrastive ($\lambda = 0.5$)	300k	353.5	0.91
strong_lb_0.05	Load Balance ($\lambda = 0.05$)	300k	197.5	0.45
strong_lb_0.1	Load Balance ($\lambda = 0.1$)	300k	360.0	0.68
strong_w_ent_0.05	Weight Entropy ($\lambda = 0.05$)	300k	464.6	0.78
strong_w_ent_0.1	Weight Entropy ($\lambda = 0.1$)	300k	254.6	0.82
strong_all_reg	Combined Strong	300k	352.9	0.89
Best Configuration (3 seeds)				
best_config_42	Gumbel+TempAnneal+Contrast+LB	300k	253.4	0.92
best_config_123	Gumbel+TempAnneal+Contrast+LB	300k	318.5	0.87
best_config_456	Gumbel+TempAnneal+Contrast+LB	300k	386.0	0.88

A.2 Computational Resources

All experiments were run on a single NVIDIA GPU (CUDA-enabled). Training times:

- 100k steps: $\sim$1 minute
- 200k steps: $\sim$2 minutes
- 300k steps: $\sim$3 minutes
- 500k steps: $\sim$5 minutes

Total compute: approximately 3 GPU-hours for all 33 models.