

Practical Approaches to Robotic Learning: Safety, Efficiency, and Adaptability

**Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
(Dr.-Ing.)**

**von der KIT-Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)**

**genehmigte
DISSERTATION**

von

M.Sc. Mohammd Karam Daaboul

geb. in Latakia, Syrien

Karlsruhe, 2026

Datum der mündlichen Prüfung: 14.07.2026
Hauptreferent: Prof. Dr.-Ing. Johann Marius Zöllner
Korreferent: Prof. Dr. Gerhard Neumann

Acknowledgments

Writing this dissertation has been a long journey, and I certainly did not walk this path alone.

First, I want to send a huge thank you to my supervisors, Professor Marius Zöllner and Professor Gerhard Neumann. Marius, thank you for giving me the freedom to explore my own ideas and for always standing by my side. Gerhard, thank you for your sharp feedback and for pushing me to think deeper. I am lucky to have had you both in my corner.

To my colleagues, you made the hard days easier. Karl Kurzer, you were a big supporter the entire time, thank you. To Philip Keller, Marcus Fechner, Tim Joseph, Sahin Tas, Philipp Stegmaier, Robert Wilbrandt and Michael Weber: thank you for the help, the laughs, and for being such a great team. To Sonja Göttl and Nicole Rösch, thank you for your support whenever I came with all my questions. I also want to thank all my colleagues at AIFB, TKS, and IDS. Being surrounded by such talented and kind people made this journey not only possible but enjoyable.

I also want to thank my students, especially Florian Kuhm, Moritz Zanger, Jan Bieser, Maximilian Schiak, and Christopher Gabrian. I learned just as much from you as you did from me. Without your hard work, I could not have achieved this.

To my friends Ada, Cemre, Max, Simon, Daniel, Lia, Franzi, Sarah, Nico and Egge: you gave me the best memories of these years. Thank you for always being there. And Julia, thank you for your love and for turning the final chapter into something beautiful.

Finally, my heart belongs to my family. Every word in this dissertation carries a piece of you. To my uncle Bashar Muselmani, you were there when it mattered most. To my sister Bana, you never stopped believing in me, even when I doubted myself. To my brother Hadi, who walked beside me when the path was still uncertain and helped me find my footing in a new country. I would not be who I am today without you.

And to my parents, Ghassan Daaboul and Basela Muselmani: you dreamed this dream first. In a small city on the Mediterranean, you chose to give your children everything you never had. And you did. Every step I have taken, every word I have written, carries your love in it. Thank you is not enough. It will never be enough. But from the bottom of my heart, thank you.

Kurzfassung

Menschen verfügen über die bemerkenswerte Fähigkeit neue Fertigkeiten schnell zu erlernen und dabei stets ihre Sicherheit zu wahren. Ein Kind, das Fahrradfahren lernt, probiert verschiedene Balance-Strategien und Geschwindigkeiten aus, vermeidet jedoch instinktiv Verhaltensweisen, die zu ernsthaften Verletzungen führen könnten. Diese Integration von Sicherheit und Lernen stellt eine zentrale Herausforderung bei der Entwicklung autonomer Roboter dar, die in der physischen Welt agieren müssen.

Reinforcement Learning hat in simulierten Umgebungen beeindruckende Ergebnisse erzielt, doch der Einsatz auf realen Robotern offenbart wesentliche Defizite. Reale Robotik erfordert drei grundlegende Fähigkeiten: das Lernen aus begrenzten Interaktionsdaten, die schnelle Anpassung an neue Bedingungen sowie die Gewährleistung von Sicherheit während der gesamten Exploration. Bestehende Ansätze behandeln diese Anforderungen meist isoliert statt als zusammenhängendes Problem.

Model-Free Reinforcement Learning bietet den Vorteil, Control Policies direkt aus Erfahrung zu erlernen, ohne auf genaue Dynamikmodelle angewiesen zu sein. Allerdings erfordert dieser Ansatz umfangreiche Interaktionen, was ihn für physische Roboter unpraktikabel macht. Model-Based Methoden begegnen diesem Problem der Sample-Effizienz, indem sie prädiktive Modelle lernen, die Planung und Rollouts in der Imagination ermöglichen und so den Bedarf an realen Daten reduzieren. Der Nachteil besteht darin, dass Modellungenauigkeiten die Policy Updates verzerren und die Leistung beeinträchtigen können.

Über die Unterscheidung zwischen Model-Based und Model-Free hinaus verfolgen zwei weitere Forschungsrichtungen ergänzende Ziele. Safe Reinforcement Learning bietet Methoden zur Constraint-Erfüllung und gewährleistet, dass Agenten während des Trainings innerhalb sicherer Grenzen operieren. Diese Methoden setzen jedoch typischerweise eine Datenverfügbarkeit voraus, die physische Systeme nicht bieten können. Meta-Learning konzentriert sich auf die schnelle Anpassung an neue Aufgaben oder Umgebungen, wobei die meisten existierenden Ansätze Sicherheitsbeschränkungen ignorieren und das System somit während der Adaptation Risiken aussetzen.

Die vorliegende Arbeit leistet vier technische Beiträge. Constrained Model-Based Policy Optimization verbindet gelernte Dynamikmodelle mit prinzipiengeleiteter Constraint-Durchsetzung und ermöglicht so sample-effizientes Policy-Lernen unter Sicherheitsanforderungen. Model-Based Meta-Adaptation with Behavioral

Constraints adressiert das Problem der Anpassung an unbekannte Dynamiken, indem die Policy darauf beschränkt wird, in der Nähe sicherer Referenzverhalten aus früherer Erfahrung zu bleiben. Constrained Model-Agnostic Meta-Learning führt einen Meta Safety Critic ein, der die Meta-Optimierung zu Initialisierungen lenkt, die sowohl schnelle Adaptation als auch konsistente Constraint-Erfüllung unterstützen. Ein vierter Beitrag befasst sich mit den rechentechnischen Herausforderungen des Vision-based Policy-Lernens, indem Off-Policy Reinforcement Learning mit Teacher-Student Distillation in massiv paralleler Simulation für die Lokomotion vierbeiniger Roboter kombiniert wird.

Umfangreiche Simulationsexperimente validieren diese Beiträge und der Einsatz einer Vision-based Locomotion Policy auf einem Unitree GO2 Roboter demonstriert die Anwendbarkeit in der realen Welt. Die Ergebnisse zeigen, dass die Behandlung von Sicherheit als grundlegendes Designprinzip, anstelle einer nachgelagerten Komponente, Algorithmen hervorbringt, die gleichzeitig sicher, sample-effizient und schnell adaptiv sind.

Abstract

Humans possess a remarkable capacity to acquire new skills rapidly while maintaining safety throughout exploration. A child learning to ride a bicycle experiments with different balancing strategies and speeds, yet instinctively avoids behaviors that risk serious injury. This integration of safety into learning poses a central challenge in developing autonomous robotic agents that must operate in the physical world.

Reinforcement learning has achieved impressive results in simulated environments, yet deploying these systems on physical robots exposes critical gaps. Real-world robotics demands three essential capabilities: learning from limited interaction data, adapting quickly to new conditions, and maintaining safety throughout exploration. Current approaches address these requirements separately rather than as an integrated problem.

Model-free reinforcement learning offers a compelling advantage by learning control policies directly from experience without requiring accurate dynamics models. However, the approach demands extensive interaction, making it impractical for physical robots. Model-based methods address this sample-efficiency problem by learning predictive models that enable planning and rollouts in imagination, thereby reducing real-world data requirements. The tradeoff is that model inaccuracies can bias policy updates and degrade performance.

Beyond this model-based versus model-free distinction, two further research directions pursue complementary objectives. Safe reinforcement learning provides frameworks for constraint satisfaction, ensuring agents remain within safety limits during training. These methods, however, typically assume data abundance that physical systems cannot provide. Meta-learning focuses on rapid adaptation when encountering novel tasks or environments, but most existing frameworks ignore safety constraints and therefore expose the system to risk during adaptation.

The thesis develops four technical contributions. Constrained Model-Based Policy Optimization integrates learned dynamics with principled constraint enforcement, enabling sample-efficient policy learning under safety requirements. Model-Based Meta-Adaptation with Behavioral Constraints addresses the problem of adapting to novel dynamics by constraining the policy to remain close to safe reference behaviors from prior experience. Constrained Model-Agnostic Meta-Learning introduces a meta safety critic that guides meta-optimization toward initializations supporting both rapid adaptation and consistent constraint satisfaction. A fourth contribution tackles the computational challenges of vision-based policy learning

Abstract

by combining off-policy reinforcement learning with teacher-student distillation in massively parallel simulation for quadrupedal locomotion.

Extensive simulated experiments validate these contributions, and deployment of a vision-based locomotion policy on a Unitree GO2 robot demonstrates real-world applicability. The results show that treating safety as a foundational design principle rather than an auxiliary component yields algorithms that are simultaneously safe, sample-efficient, and rapidly adaptive.

Contents

1	Introduction and Motivation	1
1.1	Motivation	1
1.1.1	Constrained Model-Based Policy Optimization	2
1.1.2	Model-Based Meta-Adaptation with Behavioral Constraints	3
1.1.3	Constrained Model-Agnostic Meta Learning	4
1.1.4	Scalable Off-Policy Learning for Vision-Based Locomotion	5
1.2	Dissertation Organization	5
2	Background	9
2.1	Sequential Decision Making	9
2.1.1	Markov Decision Processes	10
2.1.2	Value Functions and Policy Optimization	11
2.1.3	On-Policy and Off-Policy Learning	12
2.1.4	Partial Observability and High-Dimensional Observations	13
2.2	Safety Constraints in Reinforcement Learning	14
2.2.1	Why Rewards Alone Are Not Enough	15
2.2.2	Constrained Markov Decision Processes	15
2.3	Sample Efficiency and Adaptation	16
2.3.1	Model-Based Reinforcement Learning	17
2.3.2	Meta-Learning: Adapting Across Tasks	18
3	Constrained Model-Based Policy Optimization	21
3.1	Overview	21
3.2	Related Work	22
3.2.1	Constrained Reinforcement Learning	22
3.2.2	Model-Based Reinforcement Learning	23
3.3	Background and Problem Formulation	24
3.3.1	Constrained Markov Decision Process	24
3.3.2	Trust Region Policy Optimization	24
3.3.3	Constrained Policy Optimization	25
3.3.4	Model-Based Policy Optimization	26
3.3.5	Ensemble-Based Uncertainty Quantification	26
3.3.6	Problem Formulation	28
3.4	Method	28
3.4.1	Performance Bounds Under Model Dynamics	28
3.4.2	Three-Constraint Optimization Formulation	30
3.4.3	Ensemble-Based Uncertainty Quantification	31
3.4.4	Adaptive Real-Model Data Mixing	32

3.4.5	Uncertainty-Capped Rollout Horizons	33
3.5	Experimental Validation	34
3.5.1	Experimental Setup	34
3.5.2	Main Results	36
3.6	Discussion and Limitations	40
3.6.1	Sample Efficiency vs Safety Tradeoffs	40
3.6.2	Role of Adaptive Mechanisms	40
3.6.3	When CMBPO Succeeds and Fails	41
3.6.4	Limitations	41
4	Robotic Control Using Model-Based Meta-Adaptation	43
4.1	Overview	43
4.2	Related Work	44
4.2.1	Meta-Learning for Fast Adaptation	44
4.2.2	Model-Based Meta-Reinforcement Learning	45
4.2.3	Imitation Learning and Behavioral Specifications	46
4.2.4	Model Predictive Control with Constraints	46
4.3	Background and Problem Formulation	47
4.3.1	Meta-Learning Foundation	47
4.3.2	Model-Based Meta-Learning with Task Embeddings	47
4.3.3	Model Predictive Control	48
4.3.4	Problem Formulation	49
4.4	Method	51
4.4.1	Meta-Training with Task Embeddings	51
4.4.2	Reference Task Embeddings and Behavioral Constraints	52
4.4.3	Online Model Adaptation	54
4.4.4	Constrained Action Selection via Model Predictive Control	55
4.5	Experimental Validation	57
4.5.1	Experimental Setup	58
4.5.2	Main Results	61
4.6	Discussion and Limitations	63
4.6.1	Behavioral Appropriateness and Performance	63
4.6.2	Generalization Across Environmental Variations	63
4.6.3	Limitations	64
5	Constrained Model-Agnostic Meta Learning	65
5.1	Overview	65
5.2	Related Work	66
5.2.1	Safe Meta-Reinforcement Learning	66
5.3	Background and Problem Formulation	67
5.3.1	Gradient-Based Meta-Learning	67
5.3.2	Constrained Policy Optimization via Lagrangian Relaxation	69
5.4	Method	70
5.4.1	Constrained Meta-Learning Problem	71
5.4.2	Inner Loop: Task-Specific Constrained Optimization	73
5.4.3	Outer Loop: Meta-Policy Optimization with Safety Critic	73

5.4.4	Meta Safety Critic Training and Integration	74
5.5	Experimental Validation	76
5.5.1	Experimental Setup	76
5.5.2	Main Results	79
5.6	Discussion and Limitations	82
5.6.1	Safety–Adaptation Tradeoffs	82
5.6.2	Model Agnosticism and Generalization	83
5.6.3	Limitations	83
6	Scaling Off-Policy Learning for Practical Robotic Deployment	85
6.1	Overview	85
6.2	Related Work	86
6.2.1	Scaling Off-Policy Reinforcement Learning	86
6.2.2	Vision-Based Locomotion under Partial Observability	87
6.2.3	Asymmetric Critics and Teacher-Student Policies for Locomotion	88
6.3	Background and Problem Formulation	89
6.3.1	Twin Delayed Deep Deterministic Policy Gradient	89
6.3.2	FastTD3: Throughput-Based Scaling	90
6.3.3	Distributional Reinforcement Learning	90
6.3.4	Teacher-Student Learning and Knowledge Distillation	91
6.3.5	Problem Formulation	91
6.4	Method	92
6.4.1	Network Architecture	92
6.4.2	Hybrid Teacher-Student Training with Time-Decaying Knowledge Distillation	93
6.4.3	Action Space and Control Interface	95
6.5	Experimental Evaluation	96
6.5.1	Experimental Setup	96
6.5.2	Overall Comparison of Approaches	100
6.5.3	Performance on the Real Robot	102
6.6	Discussion and Limitations	104
6.6.1	Teacher Supervision and Off-Policy Integration	104
6.6.2	Vision Representations and Partial Observability	104
6.6.3	Limitations	105
7	Conclusion	107
7.1	Summary of Contributions	107
7.1.1	Constrained Model-Based Policy Optimization	107
7.1.2	Robotic Control Using Model-Based Meta-Adaptation	108
7.1.3	Constrained Model-Agnostic Meta Learning	108
7.1.4	Scalable Off-Policy Learning for Vision-Based Locomotion .	109
7.2	Relationships Between Contributions	109
7.2.1	Safety Across Learning Paradigms	109
7.2.2	Efficiency Through Imagination and Memory	110
7.2.3	Adaptation Mechanisms Across Timescales	110

7.3	Future Directions	111
7.3.1	Scalability to Complex Task Distributions	111
7.3.2	Real-World Deployment and Continual Learning	111
7.4	Concluding Remarks	112
Appendix		113
A	Appendix to Chapter 3	115
A.1	Proof of Theorem 3.1: Return difference under model uncertainty .	115
A.2	CPO step details	117
A.3	Hyperparameters	118
B	Appendix to Chapter 4	121
B.1	Hyperparameters	121
C	Appendix to Chapter 5	125
C.1	Hyperparameters	125
D	Appendix to Chapter 6	129
D.1	Visual Augmentation and Normalization	129
D.2	Recurrent Off-Policy Sequence Training	129
D.3	Hyperparameters	130
List of Figures		135
List of Tables		137
Own Publications		139
Student Work		140
Bibliography		141

1 Introduction and Motivation

“Yesterday I was clever, so I wanted to change the world. Today I am wise, so I am changing myself.”

— Rumi

1.1 Motivation

Humans have a remarkable ability to learn new skills quickly while maintaining safety throughout the learning process. A child learning to ride a bicycle explores different balancing strategies and speeds, yet instinctively avoids behaviors that could cause serious injury. A person learning to pour water into a glass experiments with different angles and speeds, but does not deliberately pour onto electrical outlets. This simultaneous pursuit of learning and safety appears effortless, but it represents one of the most challenging problems in robotic learning.

Standard reinforcement learning follows a simple recipe. Define a task and train a policy from scratch through exploration. The agent tries different actions, observes the outcomes, and gradually discovers which behaviors yield the highest rewards. This approach has yielded remarkable results in simulations, including mastering Atari games, achieving superhuman performance in the game of Go, and learning complex manipulation skills. However, when we deploy these same methods on physical robots, we encounter a fundamental problem: *exploration is dangerous*.

A robot learning to walk might try falling to see what happens. A manipulator learning to grasp might try smashing objects to understand fragility. Unlike simulation, where we simply reset the environment, real-world failures have real consequences, including broken hardware, damaged environments, and potential harm to humans. The trial-and-error paradigm that works so well in simulation becomes fundamentally incompatible with physical systems where even a single catastrophic failure can end the learning process.

The traditional solution has been conservative: hand-designing safe controllers, restricting exploration to known-safe regions, or relying heavily on human demonstrations to avoid dangerous states. Nevertheless, this conservative approach severely limits capability. Robots become unable to adapt when environments change, fail in conditions slightly different from training, and require extensive human effort for each new deployment. We end up with systems that work in

controlled laboratory settings but fail in the unpredictable real world, not due to a lack of capability, but because we have constrained their ability to learn and adapt out of safety concerns.

This creates a fundamental tension: we need robots that can explore and adapt to succeed in diverse real-world environments, yet we also need robots that maintain safety during exploration and adaptation. Must we forever choose between adaptive robots that are unsafe and safe robots that cannot adapt?

This dissertation argues that the answer is no. The key insight is to change *what* robots learn. Instead of learning individual tasks from scratch, robots should learn *how to learn safely*. Rather than optimizing for performance in a single specific environment, robots should prioritize the ability to quickly adapt to new environments while maintaining safety throughout the adaptation process. This is not merely a shift in training data; it represents a fundamental shift in the learning objective itself.

Existing approaches treat safety, efficiency, and adaptation as separate concerns, often in tension with one another. Safe reinforcement learning adds constraints but struggles with exploration. Model-based methods improve sample efficiency but introduce compounding prediction errors that corrupt safety estimates. Meta-learning enables rapid adaptation but prioritizes task performance over behavioral appropriateness. What is missing is a unified framework that treats safety not as an afterthought, but as a fundamental component integrated throughout the learning process.

This dissertation develops such a framework through four complementary contributions. Each addresses a distinct challenge in practical robotic learning: achieving sample efficiency under safety constraints when models are imperfect, adapting quickly without compromising desirable behaviors, meta-learning across diverse tasks while maintaining safety during adaptation, and scaling vision-based methods for real-world deployments. Together, they demonstrate that safety, efficiency, and adaptability need not be in competition when constraints are treated as fundamental design elements rather than afterthoughts.

1.1.1 Constrained Model-Based Policy Optimization

Can we leverage learned models to gain sample efficiency while maintaining safety when model errors can violate constraints?

Model-based reinforcement learning is attractive in robotics. A learned dynamics model lets the robot simulate futures and reduce real interactions. But even small one-step errors accumulate over long rollouts. In safety-critical settings, this compounding error is not only a matter of reduced performance; it can cause the planner to label a sequence of actions as safe when, on the real system, it violates constraints.

This dissertation introduces a model-based reinforcement learning framework that couples ensemble-based uncertainty estimation with constrained policy optimization. The framework explicitly controls the trade-off between data efficiency and safety in the presence of model errors. The core idea is simple: disagreement within an ensemble of dynamics models (quantified through prediction variance) serves as a measure of epistemic uncertainty. This signal determines how much to trust the model versus relying on real data. Rather than using fixed rollout lengths or constant data mixing ratios, the algorithm adapts both online based on the uncertainty signal.

When uncertainty is low, the algorithm uses longer model rollouts with more synthetic data, which improves data efficiency. When uncertainty spikes (indicating poorly explored regions or distribution shift), the algorithm shortens the rollouts and relies more heavily on real data for policy updates. This prevents the model from confidently predicting nonsense in areas it has not seen much. By explicitly tracking uncertainty and feeding it into constrained policy updates, the approach achieves the data efficiency of model-based methods without sacrificing the robustness expected from model-free approaches.

This contribution takes the form of the Constrained Model-Based Policy Optimization (CMBPO) algorithm, originally presented at IROS 2021 (Zanger et al. 2021). CMBPO provides a foundation for safe and sample-efficient robotic learning within the broader framework developed in this dissertation.

1.1.2 Model-Based Meta-Adaptation with Behavioral Constraints

Can we adapt quickly to new environments while preserving validated movement patterns and safety-relevant behaviors?

Robots deployed in diverse environments must adapt their behavior to new conditions such as changes in friction, payload, or terrain. Meta-learning offers a natural approach by learning an adaptation mechanism from a distribution of tasks, so that only a small amount of new experience is required in each environment. However, standard meta-learning methods focus almost exclusively on post-adaptation performance. They adapt models and policies aggressively to optimize reward and often ignore whether the adapted behavior remains similar to previously validated motions. In practice, this can produce gaits that achieve higher returns but are qualitatively undesirable or even unsafe compared with known stable behaviors.

A model-based meta-adaptation framework addresses this gap by encoding behavioral preferences directly into the planning process. The method maintains an embedding-conditioned dynamics model that continuously adapts based on recent observations. During planning, this adapted model is queried with two task embeddings: an adaptive embedding that tracks the current environment

and a reference embedding associated with a task where appropriate behavior has been validated. The core mechanism treats the discrepancy between trajectories predicted under these two embeddings as a measure of behavioral deviation and penalizes candidate actions that induce large deviations while still optimizing task reward.

When the environment changes, the adaptive embedding adjusts to capture new dynamics while the reference embedding remains fixed. The planner compares predicted state transitions under both embeddings at each step. Actions that produce trajectories similar to the reference behavior receive higher scores, biasing the controller toward validated movement patterns even as it adapts to new conditions.

This contribution appears as the Model-Based Meta-Adaptation Controller (MAC), originally presented at ICRA 2023 (Daaboul et al. 2023). MAC demonstrates how fast adaptation and behavioral safety can be achieved simultaneously through learned task representations.

1.1.3 Constrained Model-Agnostic Meta Learning

Can we adapt quickly across many tasks while staying within shared safety constraints during both adaptation and deployment?

Robots that operate in the real world rarely face a single task. They must specialize to new goals or environment configurations quickly, using limited data, while still respecting shared safety constraints such as collision costs or torque limits. Gradient-based meta-reinforcement learning methods, such as Model-Agnostic Meta-Learning (MAML) (Finn et al. 2017), address the adaptation speed problem by learning initial policy parameters that can be fine-tuned with only a few gradient steps on a new task. However, standard MAML optimizes only for post-adaptation return and ignores safety, so the resulting initializations may adapt quickly but rely on unsafe trajectories during adaptation or when the meta-policy is deployed directly.

Integrating safety directly into the meta-objective requires a constrained meta-learning formulation. The approach formulates meta-learning as a constrained bi-level optimization problem where adaptation performance and constraint satisfaction are optimized jointly. The inner loop applies constrained policy optimization to obtain task-specific policies satisfying prescribed cost bounds after a fixed number of adaptation steps. The outer loop, guided by a learned safety critic, updates meta-parameters to minimize expected cost across the task distribution.

This contribution shows that fast adaptation and safety are compatible objectives when constraint satisfaction is integrated into the meta-learning process. The approach is implemented as the Constrained Model-Agnostic Meta-Learning (CMAML) algorithm, initially presented at the CoRL 2024 workshop (Daaboul et al. 2024).

1.1.4 Scalable Off-Policy Learning for Vision-Based Locomotion

Can we make vision-based off-policy learning practical for real robots despite limited visual parallelism?

Off-policy reinforcement learning is attractive for robotics because it reuses experience efficiently and updates policies asynchronously from large replay buffers. Modern methods exploit this by running thousands of parallel simulations to generate diverse data continuously, keeping replay buffers fresh and value estimates stable. This throughput-based approach works well for proprioceptive control where observations are low-dimensional and many environments fit in GPU memory. Vision-based control breaks this scaling strategy. Visual encoders consume substantial memory, drastically reducing feasible parallelism. Fewer environments mean less diverse batches and stronger temporal correlations in replay buffers. High update-to-data ratios then cause critics to develop overconfident value estimates from repeatedly processing the same limited visual data, leading to training instability and policies that fail to transfer to hardware.

Architectural stabilization and behavioral grounding compensate for limited parallelism when simulation throughput alone cannot maintain stability. The approach adopts hyperspherical normalization to constrain feature magnitudes under non-stationary replay data, uses recurrent encoders to aggregate temporal context from egocentric depth sequences, and introduces a hybrid training objective that combines off-policy reinforcement learning with time-decaying distillation from a privileged teacher. Early in training, when visual experience is limited and critic estimates are unreliable, strong teacher supervision prevents the policy from exploiting poorly calibrated values. As the replay buffer accumulates diverse visual transitions and value estimates mature, the distillation weight decays exponentially and control gradually transfers to the reinforcement learning objective.

This contribution demonstrates that off-policy learning remains viable for vision-based locomotion when architectural choices and training objectives compensate for reduced parallelism. The complete framework and hardware deployment results are currently under review for publication.

1.2 Dissertation Organization

The remainder of this dissertation is organized as follows:

Chapter 2 establishes the mathematical foundations for safe, efficient, and adaptive robotic learning. We define the core decision-making frameworks (Markov Decision Processes, Constrained MDPs, and meta-learning formulations).

Chapter 3 presents Constrained Model-Based Policy Optimization, demonstrating how ensemble-based uncertainty quantification enables sample-efficient safe exploration through adaptive model utilization.

Chapter 4 introduces Model-Based Meta-Adaptation, showing how reference behavior constraints enable rapid adaptation to new dynamics while preserving preferred movement patterns.

Chapter 5 develops Constrained Model-Agnostic Meta-Learning, extending gradient-based meta-learning to incorporate safety constraints through dual-level optimization and meta safety critics.

Chapter 6 addresses scalable vision-based learning, developing techniques for stable high-throughput training from parallel simulation with visual observations.

Chapter 7 concludes with a discussion of limitations, open questions, and future directions, including pathways toward real-world deployment and extensions to more complex multi-agent settings.

Together, these contributions provide a comprehensive framework for practical robotic learning that addresses the fundamental challenges of safety, efficiency, and adaptability. Rather than treating these as competing objectives, we demonstrate how they can be achieved simultaneously through principled algorithm design that integrates safety throughout the learning process rather than relegating it to post-hoc verification.

Contribution	Key Problem Addressed	Safety	Efficiency	Adaptability
Ch. 3: CMBPO	Leveraging learned models for efficiency while model errors could violate constraints.	✓ (via constrained optimization)	✓ (via model-based learning)	
Ch. 4: MAC	Adapting quickly to new dynamics while preserving preferred movement and safety behaviors.	✓ (via behavioral constraints)	✓ (via model-based learning)	✓ (via meta-learning)
Ch. 5: C-MAML	Meta-learning across tasks with varying objectives, ensuring safety during and after adaptation.	✓ (via meta-safety critic)		✓ (via meta-learning)
Ch. 6: Scaling	Making vision-based off-policy learning stable and practical for real robot deployment.		✓ (via scalable off-policy)	

Table 1.1: Mapping of Dissertation Contributions to Core Research Themes

2 Background

This dissertation addresses a fundamental question in robotic learning: *How can robots learn safe and effective behaviors efficiently while adapting to changing environments?* Answering this question requires formal frameworks that precisely characterize the interplay between safety, efficiency, and adaptation: the three pillars of practical robotic learning systems.

Unlike simulation environments where exploration is essentially cost-free, physical robotic systems face hard constraints. Hardware damage, safety violations, and operational limits cannot be treated as soft penalties. Similarly, while model-free Reinforcement Learning (RL) can achieve impressive results given millions of interactions, real robots must learn from limited data (hundreds or thousands of interactions, not millions). Finally, robots deployed in the real world encounter diverse and changing conditions, requiring rapid adaptation without compromising safety or relearning from scratch.

This chapter establishes the mathematical foundations for reasoning about these problems. Rather than providing comprehensive algorithmic details (which are presented in Chapters 3-6, where they are directly applied), we focus on defining the core decision-making frameworks that enable safe, efficient, and adaptive robotic learning. Section 2.1 introduces the fundamental concepts of sequential decision making, including Markov Decision Processes, value functions, policy optimization methods, and partial observability. Section 2.2 formalizes safety requirements through Constrained Markov Decision Processes. Section 2.3 presents model-based reinforcement learning and meta-learning as mechanisms for achieving sample efficiency and rapid adaptation.

2.1 Sequential Decision Making

Robots learn by repeatedly interacting with their environment over time. Each interaction provides information: what happened after taking an action, whether the outcome was desirable, and what new situations emerged. To reason about this learning process mathematically, we need formal structures that capture how actions lead to outcomes, how outcomes accumulate into long-term performance, and how agents can learn from diverse data sources. We begin by formalizing the interaction cycle through Markov Decision Processes, then introduce value functions that quantify the quality of states and actions. We examine how algorithms differ based on whether they learn from current policy data (on-policy)

or reuse past experience (off-policy), and address the practical reality that robots observe the world through sensors rather than having access to complete state information.

2.1.1 Markov Decision Processes

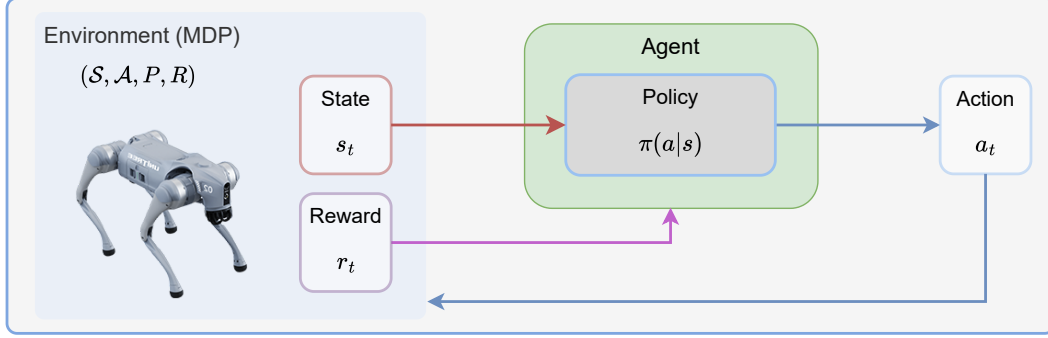


Figure 2.1: Markov decision process framework: The agent observes the current state s_t from the environment and uses its policy $\pi(a|s)$ to select an action a_t . The environment, characterized by the tuple $(\mathcal{S}, \mathcal{A}, P, R)$, receives the action and returns both a reward r_t and the next state s_{t+1} according to the transition dynamics P . This interaction loop continues, with the agent’s objective being to learn a policy that maximizes the expected cumulative discounted reward over time.

We formalize robotic control as a discrete-time sequential decision-making problem through the framework of Markov decision processes. A Markov Decision Process (MDP) is characterized by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ the transition dynamics, $\mathcal{R}: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function, and $\gamma \in [0, 1]$ the discount factor (Sutton and Barto 2018). The defining Markov property ensures that transitions depend only on the current state and action, enabling tractable learning algorithms.

At each time step t , the agent observes state $s_t \in \mathcal{S}$ and selects action $a_t \in \mathcal{A}$ according to its policy $\pi: \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$. The environment responds by providing reward r_t and transitioning to the next state s_{t+1} sampled from $P(\cdot | s_t, a_t)$. This interaction generates a trajectory τ

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots),$$

representing the complete sequence of states, actions, and rewards experienced during execution. Figure 2.1 illustrates this cyclical agent-environment interaction.

The objective in reinforcement learning is to find a policy that maximizes the expected discounted return

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right], \quad (2.1)$$

where the discount factor γ balances immediate versus future rewards. Throughout this work, we write $J(\pi)$ when the policy is clear from context.

2.1.2 Value Functions and Policy Optimization

Finding optimal policies requires a principled way to measure the quality of states and actions. We formalize this through value functions, which provide numerical estimates of expected returns.

The state value function V^π quantifies the expected cumulative reward when starting from state s and following policy π :

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \right] \quad (2.2)$$

The action-value function Q^π extends this to state-action pairs, measuring the expected return from taking action a in state s and subsequently following policy π :

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right] \quad (2.3)$$

These functions satisfy a recursive relationship through the Bellman equation:

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P(\cdot \mid s, a)} [r(s, a) + \gamma V^\pi(s')] \quad (2.4)$$

This decomposition reveals a fundamental structure: the value of an action equals the immediate reward plus the expected value of the successor state. This recursive characterization underlies most reinforcement learning algorithms (Sutton and Barto 2018).

The advantage function A^π measures how much better an action is compared to the average action under the current policy:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \quad (2.5)$$

In practice, advantages are estimated using methods like Generalized Advantage Estimation (GAE) (Schulman et al. 2016), which balances bias and variance in the estimation.

2 Background

For robotic systems with continuous states and actions, computing these values exactly is intractable. The state space is simply too large. Instead, we approximate value functions using neural networks. Actor-critic methods separate this approximation into two components (Figure 2.2): an actor $\pi_\theta(a|s)$ parameterized by θ that chooses actions, and a critic $Q_\psi(s, a)$ parameterized by ψ that evaluates those choices. The critic learns to predict long-term outcomes using the Bellman structure, while the actor adjusts its behavior based on the critic's assessments. A critical question remains: what data should we use to train these components?

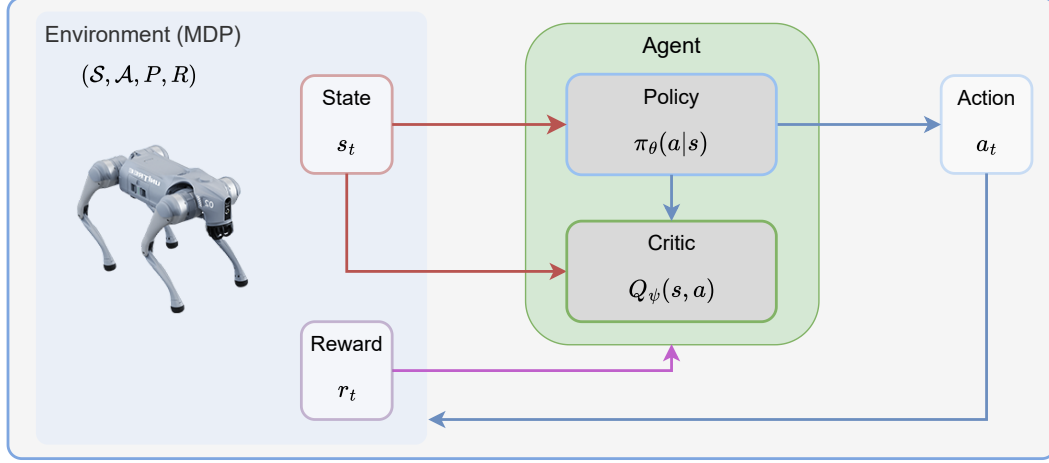


Figure 2.2: Actor-critic architecture: The agent maintains two coupled components. The actor, parameterized by θ , implements a stochastic policy $\pi_\theta(a|s)$ that selects actions given the current state. The critic, parameterized by ψ , estimates the action value function $Q_\psi(s, a)$ from experience tuples (s_t, a_t, r_t, s_{t+1}) . The critic is trained to predict discounted returns, and the actor is updated to choose actions that receive high value under the critic while still exploring new behaviors.

2.1.3 On-Policy and Off-Policy Learning

A fundamental distinction in reinforcement learning is whether algorithms learn from data generated by the policy being optimized (on-policy) or from data generated by other policies (off-policy).

On-Policy Learning. On-policy methods optimize a policy π using only trajectories τ sampled from that same policy. The policy gradient updates the parameters θ in the direction:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t) A^{\pi_\theta}(s_t, a_t) \right] \quad (2.6)$$

where A^{π_θ} is the advantage function under policy π_θ . Critically, this expectation is over trajectories generated by the current policy π_θ . After each policy update

that changes θ , the data distribution changes, requiring fresh data collection. This constraint severely limits sample efficiency, as collected data is typically used once and then discarded.

Off-Policy Learning. Off-policy methods relax this requirement by learning from data generated by any policy. The key enabling mechanism is the Bellman equation for Q-functions, which allows us to learn action values from any transition regardless of how it was generated. Given a transition (s, a, r, s') from any behavior policy, we can update our Q-function estimate:

$$Q_\psi(s, a) \leftarrow Q_\psi(s, a) + \alpha \left[r + \gamma \max_{a'} Q_\psi(s', a') - Q_\psi(s, a) \right] \quad (2.7)$$

This temporal difference update is valid for any transition, enabling learning from a replay buffer $\mathcal{D}$ that stores past transitions (s, a, r, s') from many different policies. During training, we sample mini-batches from $\mathcal{D}$ to perform gradient updates, using each experience multiple times. For robotics, where real-world data is expensive, this reuse can dramatically improve sample efficiency.

However, off-policy learning introduces challenges. The replay buffer contains data from old policies, which may have potentially different behavior than the current policy. This distribution mismatch can cause value function estimates to diverge, particularly when the behavior policy differs significantly from the current policy. Modern off-policy algorithms like Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al. 2018) and Soft Actor-Critic (SAC) (Haarnoja et al. 2018) address these challenges through careful design choices: target networks for stability, clipped double Q-learning to prevent overestimation, and delayed policy updates to allow value functions to converge.

2.1.4 Partial Observability and High-Dimensional Observations

In practice, robots rarely have access to the full state of the environment. A quadruped does not know the exact friction coefficient of the terrain it is walking on. A manipulator does not know the precise mass distribution of an object it is grasping. Instead, robots observe the world through various sensors, including cameras, depth sensors, joint encoders, and force-torque sensors.

We formalize this through the partially observable Markov decision process (POMDP) framework. At each time step, rather than observing the true state $s_t \in \mathcal{S}$, the robot receives an observation $o_t \in \mathcal{O}$ from an observation space $\mathcal{O}$, generated by an observation function:

$$o_t = h(s_t, \epsilon_t) \quad (2.8)$$

where $h : \mathcal{S} \times \mathcal{E} \rightarrow \mathcal{O}$ maps states to observations and $\epsilon_t \in \mathcal{E}$ represents stochastic sensor noise. Since individual observations may not capture all decision-relevant information, the policy must condition on observation histories: $\pi(a_t | o_0, o_1, \dots, o_t)$, where we denote $o_{\leq t} = (o_0, o_1, \dots, o_t)$ as the observation history up to time t .

2 Background

This partial observability becomes particularly challenging when observations are high-dimensional. Learning from visual observations, such as camera images, introduces several complications. Sequential camera frames are highly correlated, violating standard RL independence assumptions and complicating the temporal credit assignment process. Determining which visual features are relevant for decision-making is challenging when the observation space contains thousands of pixels, but only a small subset is important for the task. Moreover, learning directly from pixels typically requires significantly more data than learning from low-dimensional state representations.

Recurrent policies for history encoding. When individual observations provide incomplete information, we can recover approximately Markovian representations by conditioning on observation histories. Recurrent neural networks accomplish this by maintaining a hidden state $h_t \in \mathcal{H}$ that summarizes past observations:

$$h_t = f_{\text{RNN}}(o_t, h_{t-1}; \xi) \quad (2.9)$$

$$\pi(a_t | o_{\leq t}) \approx \pi_{\theta}(a_t | h_t) \quad (2.10)$$

where f_{RNN} is a recurrent function parameterized by ξ (often included within the policy parameters θ), and $\mathcal{H}$ denotes the hidden state space. Common architectures include Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU), which learn which aspects of history to retain and which to forget. For vision-based locomotion, a robot might aggregate the last several depth frames through a GRU to infer terrain geometry and estimate its velocity, information not available in a single observation.

Asymmetric actor-critic architectures. In simulation, we often have access to privileged information unavailable to deployed robots. Terrain height maps, object masses, and exact friction coefficients can be queried from the simulator but not measured by sensors. Asymmetric actor-critic methods exploit this during training: the critic $Q_{\psi}(s, a)$ evaluates actions using full state information s , while the policy $\pi_{\theta}(a|o)$ learns from realistic observations o . This configuration provides the policy with informative learning signals without creating a dependency on privileged information at deployment. The critic learns more accurate value estimates from the complete state, guiding the policy to discover behaviors that work from limited observations alone. This pattern proves particularly effective for vision-based control, where critics leverage geometric information while policies learn from egocentric depth or RGB images.

2.2 Safety Constraints in Reinforcement Learning

Standard reinforcement learning optimizes policies to maximize expected reward. For physical robots, this objective proves insufficient when specific actions can cause irreversible damage. We begin by examining why reward-based objectives alone cannot capture safety requirements, then formalize safety through

Constrained Markov Decision Processes that augment the standard MDP with explicit cost functions and constraint thresholds. Just as we approximate value functions with critics (Section 2.1), we will later approximate expected costs using learned cost critics that enable practical implementation of these constrained formulations.

2.2.1 Why Rewards Alone Are Not Enough

In simulation, maximizing reward works well. An agent learning to play Atari can try millions of actions, fail frequently, and eventually discover good strategies. However, physical robots operate under different constraints. A robot learning to walk cannot simply try falling over to see what happens; the hardware might break. A robotic arm learning to grasp cannot try smashing objects.

The traditional RL paradigm of "try, fail, learn, repeat" is fundamentally incompatible with physical systems where even a single failure can be catastrophic. We need a framework that explicitly reasons about safety, not as something we hope will emerge from maximizing rewards, but as a hard constraint that the learning process must satisfy.

2.2.2 Constrained Markov Decision Processes

This motivates the Constrained Markov Decision Process (CMDP), which augments the standard MDP with explicit safety constraints. In addition to rewards, we define m cost functions $C_j(s, a): \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_+$ for $j \in \{1, \dots, m\}$ that measure safety-relevant quantities. For a walking robot, these costs might capture collisions, joint torque violations, or instability. For a robotic arm, costs measure contact forces exceeding safe limits or proximity to obstacles. At each time step t , the environment returns both a reward r_t and costs $c_{j,t} = C_j(s_t, a_t)$ for each constraint.

Each cost function has a threshold d_j defining the maximum acceptable cumulative cost. The constrained optimization problem becomes:

$$\max_{\pi} J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \quad (2.11)$$

$$\text{subject to: } J_{C_j}(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t c_{j,t} \right] \leq d_j, \quad \forall j \in \{1, \dots, m\} \quad (2.12)$$

where $J_{C_j}(\pi)$ denotes the expected cost return for constraint j . The key distinction from standard MDPs is that we must satisfy constraints (2.12) while maximizing (2.11). A policy that achieves high reward but violates constraints is unacceptable, even if the constraint violations are rare. This constrained optimization problem

2 Background

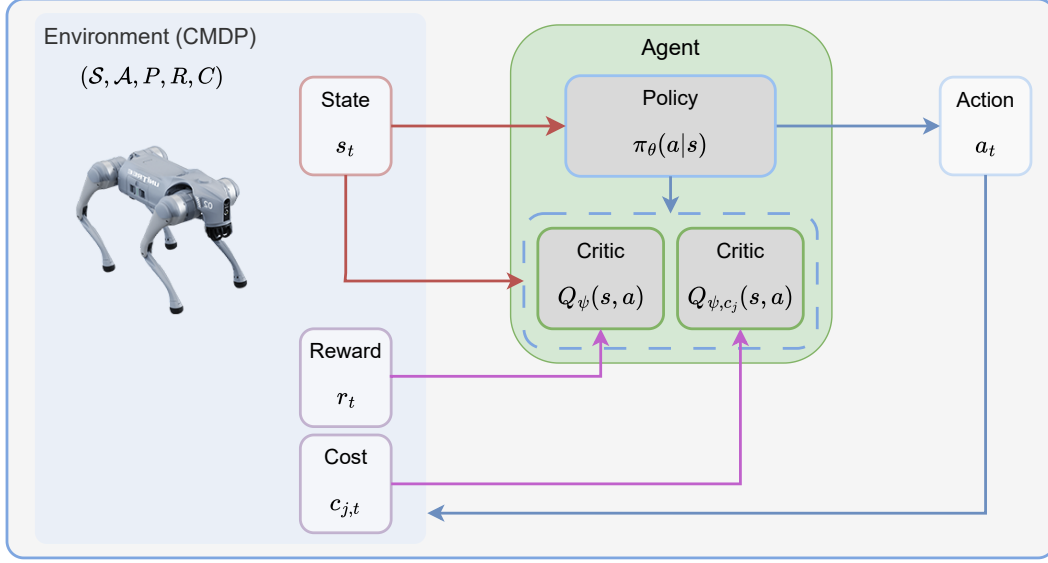


Figure 2.3: Constrained actor-critic architecture: In addition to the standard value critic $Q_\psi(s, a)$, the agent maintains cost critics $Q_{\psi, c_j}(s, a)$ that estimate expected constraint violations for different cost signals $c_{j,t}$. The environment returns both rewards r_t and costs $c_{j,t}$, enabling the policy $\pi_\theta(a|s)$ to balance reward maximization with satisfaction of multiple safety constraints. For clarity, the diagram illustrates the case with a single constraint ($j = 1$), although the framework directly generalizes to multiple constraints.

can be approached using Lagrangian methods (Boyd and Vandenberghe 2004) or trust region approaches (Achiam et al. 2017).

Similar to how we approximate action values with a critic $Q_\psi(s, a)$ (Section 2.1), we can approximate expected costs with cost critics. For each constraint j , we maintain a cost critic $Q_{\psi, c_j}(s, a)$ parameterized by ψ that estimates the expected cumulative cost:

$$Q_{\psi, c_j}(s, a) \approx \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t c_{j,t} \mid s_0 = s, a_0 = a \right] \quad (2.13)$$

This enables practical implementation of constrained reinforcement learning algorithms by providing gradient information about constraint satisfaction for each constraint type, analogous to how value critics enable policy optimization (Figure 2.3).

2.3 Sample Efficiency and Adaptation

Optimizing policies under constraints addresses what robots should do, but leaves open the question of how they learn to do it efficiently and adapt when their

world changes. Two complementary mechanisms prove essential for practical robotic learning. Model-based reinforcement learning learns dynamics models that approximate how the world responds to actions, then uses these models to simulate experience without physical execution. Meta-learning operates at a different level entirely, optimizing across distributions of tasks to find policy initializations that adapt quickly with minimal data.

2.3.1 Model-Based Reinforcement Learning

Sample efficiency is the central bottleneck in applying reinforcement learning to physical robots. Model-free algorithms learn purely from observed outcomes, requiring extensive interaction to discover effective behaviors. For physical robots, collecting millions of samples is impractical due to time constraints, hardware wear, and safety considerations.

Model-based reinforcement learning addresses this by learning an explicit representation of environment dynamics. The foundational idea originates from Dyna (Sutton and Barto 2018), which demonstrated how learned models can generate synthetic experiences to augment real data. At its core, this approach learns a parametric dynamics model $P_\phi : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$, parameterized by ϕ , that predicts the distribution over next states given the current state and action, approximating the true transition dynamics P from the MDP.

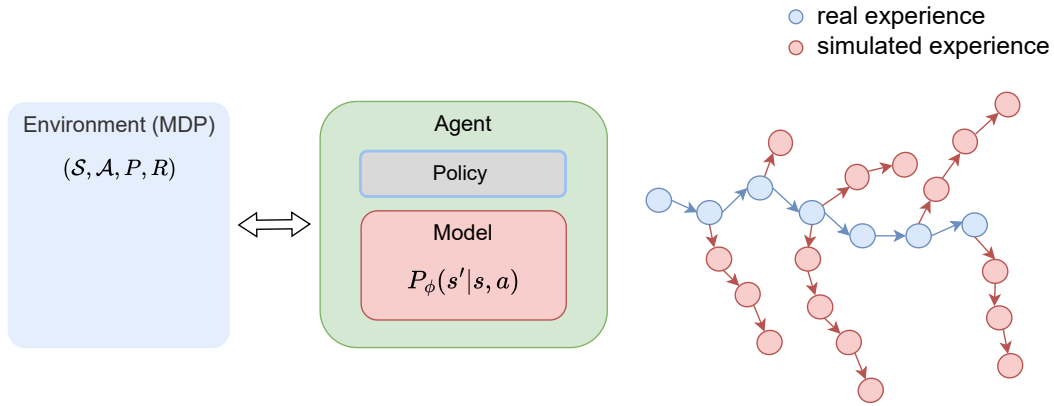


Figure 2.4: Model-based reinforcement learning framework: The agent interacts with the underlying Markov decision process $(\mathcal{S}, \mathcal{A}, P, R)$ and collects experience transitions (s_t, a_t, r_t, s_{t+1}) . From this data, the agent learns a parametric dynamics model P_ϕ that approximates the true transition kernel P . The learned model can then be used to generate synthetic rollouts or to plan over imagined trajectories, providing additional training signal for the policy and critic and substantially improving sample efficiency.

2 Background

With this learned model, we can generate synthetic experience without physical execution (Figure 2.4). Given a current state s_0 , we simulate rollouts of horizon length H by repeatedly sampling from the learned dynamics:

$$s_{t+1} \sim P_\phi(\cdot | s_t, a_t), \quad t = 0, 1, \dots, H - 1 \quad (2.14)$$

with rewards computed using the known reward function $r_t = R(s_t, a_t)$. These imagined trajectories $(s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ can train policies just as real experience does. The sample efficiency improvement is substantial: each real transition can generate many synthetic transitions through branching rollouts, multiplying the effective size of our dataset.

However, learned models are imperfect. Even small one-step prediction errors accumulate across multi-step rollouts. If ϵ_t denotes the prediction error at step t , errors can compound multiplicatively across the horizon, causing simulated trajectories to diverge from what would actually happen (Janner et al. 2019). This creates a fundamental tradeoff: longer horizons generate more synthetic data but with decreasing reliability, while shorter horizons maintain accuracy but provide less simulated experience per real sample.

2.3.2 Meta-Learning: Adapting Across Tasks

Real-world robots encounter varying conditions. A quadruped might need to walk on various surfaces, including grass, gravel, stairs, or ice. A manipulator might interact with objects of different weights and frictions. Retraining from scratch for each new condition is impractical; we need robots that can adapt quickly to new conditions.

This motivates meta-learning (Finn et al. 2017), or "learning to learn." Instead of learning a single task, we learn across a distribution of tasks $\mathcal{T}$, where each task $\mathcal{T}_i \sim p(\mathcal{T})$ is an MDP or CMDP with potentially different dynamics, rewards, or constraints. The goal is to learn a shared initialization θ that enables rapid adaptation to new tasks from this distribution.

The meta-learning process works as follows. During meta-training, we sample a batch of tasks from $\mathcal{T}$. For each task $\mathcal{T}_i$, we collect a small dataset $\mathcal{D}_i = \{(s, a, r, s')\}$ of transitions and adapt the policy using K steps of gradient descent with learning rate α :

$$\theta_i^{(k+1)} = \theta_i^{(k)} - \alpha \nabla_{\theta_i} \mathcal{L}_i(\theta_i^{(k)}), \quad k = 0, 1, \dots, K - 1 \quad (2.15)$$

starting from the meta-parameters $\theta_i^{(0)} = \theta$, where $\mathcal{L}_i(\theta)$ is a task-specific loss function (typically the negative expected return or policy gradient objective). The superscript (k) denotes the adaptation step. The meta-objective optimizes θ for post-adaptation performance across the task distribution:

$$\min_{\theta} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\mathcal{L}_i(\theta_i^{(K)}) \right] \quad (2.16)$$

The intuition is straightforward: rather than learning task-specific knowledge, we learn how to learn. The initialization θ encodes inductive biases that facilitate rapid fine-tuning. After meta-training, the agent can adapt to new tasks with minimal data, analogous to how humans leverage prior experience when learning new skills.

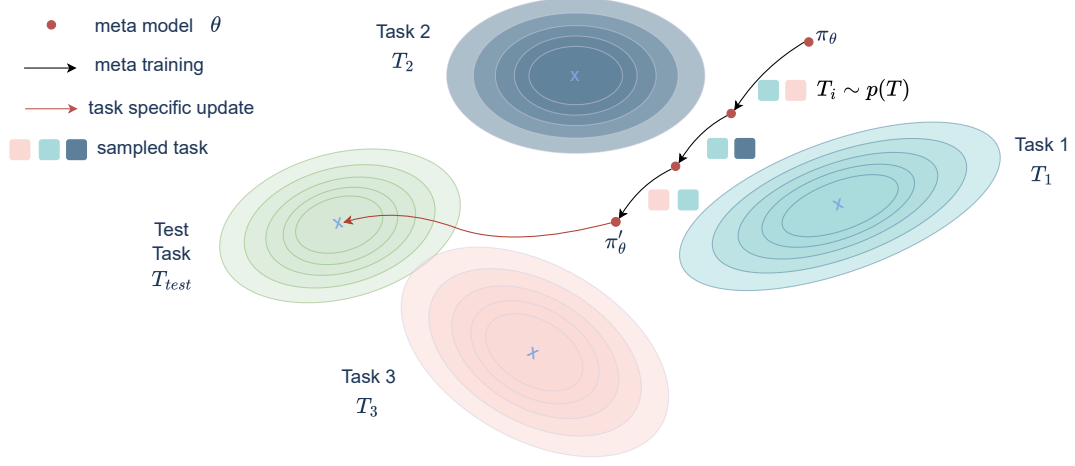


Figure 2.5: Meta-learning framework: The meta-model with parameters θ is trained across a distribution of tasks $T_i \sim p(\mathcal{T})$, illustrated as different colored task manifolds. For each sampled task, the agent performs an inner adaptation loop that updates task-specific parameters θ_i using data collected in that task. The outer meta-objective then updates the shared initialization θ so that, after only a few adaptation steps, the adapted parameters θ_i achieve high performance on their respective tasks, yielding rapid post-adaptation performance across the full task distribution.

3 Constrained Model–Based Policy Optimization

3.1 Overview

This chapter is based on work originally published at IROS 2021 (Zanger et al. 2021). © 2021 IEEE.

Deploying reinforcement learning in safety-critical robotics requires policies that learn with minimal real-world interactions and consistently respect safety constraints. Model-free safe reinforcement learning methods, such as Constrained Policy Optimization (CPO), provide principled guarantees on constraint satisfaction; however, they require large amounts of real-world experience (Achiam et al. 2017). Model-based reinforcement learning improves sample efficiency by learning dynamics models and training from synthetic rollouts (Deisenroth and Rasmussen 2011, Nagabandi et al. 2018). However, most approaches either ignore safety or rely on fixed, conservative rules that limit learning efficiency. Recent progress in model-based policy optimization has shown that short model rollouts combined with replay can match model-free performance with significantly less data (Janner et al. 2019), although safety remains an open challenge. This chapter introduces Constrained Model-Based Policy Optimization (CMBPO), a framework that combines model-based learning with constrained policy optimization. The central idea is to compute reward and cost advantage functions under learned model dynamics while explicitly managing model uncertainty so that estimates used for constraint satisfaction remain valid. In effect, CMBPO seeks the sample efficiency of model-based training and the reliability of constrained optimization in a single procedure.

We organize the method around three requirements. First, sample efficiency is achieved by computing the advantage function with model-generated rollouts starting from replay states and reusing all available experience. Second, constraint satisfaction optimizes policy updates with a constrained trust region, ensuring estimated costs remain below the specified limits. Third, uncertainty management involves bound reliance on model predictions, utilizing an ensemble-based measure of disagreement that signals when the model is unreliable (Lakshminarayanan et al. 2017, Chua et al. 2018). Concretely, CMBPO integrates three mechanisms that address these requirements: expected pairwise Kullback-Leibler (KL) disagreement from a probabilistic ensemble to quantify uncertainty, adaptive

mixing of real and model data that adjusts with uncertainty, and dynamic rollout horizons that stop trajectories when cumulative uncertainty becomes too high.

This chapter is organized as follows. Section 3.2 discusses related work in safe reinforcement learning, model-based methods, and uncertainty quantification. Section 3.3 establishes the problem formulation and necessary background. Section 3.4 presents the CMBPO algorithm in detail, including the constrained trust region formulation with model-based advantages, uncertainty quantification, and adaptive model utilization mechanisms. Section 3.5 presents experimental validation across diverse constrained control tasks. Finally, Section 3.6 concludes with a discussion of limitations and future directions.

3.2 Related Work

3.2.1 Constrained Reinforcement Learning

Safe reinforcement learning establishes theoretical foundations through CMDP (Altman 1999), which extends standard MDPs by incorporating explicit constraints on expected cumulative costs. The agent seeks to maximize the expected reward, subject to maintaining the expected costs below specified thresholds. This framework enables the development of diverse algorithmic approaches to constraint satisfaction, each offering different trade-offs between safety guarantees, computational efficiency, and sample complexity.

Trust region methods extended to handle constraints provide principled policy updates with theoretical guarantees on both performance improvement and constraint satisfaction (Achiam et al. 2017). Lagrangian approaches (Chow et al. 2018b, Ray et al. 2019) transform constrained problems into unconstrained optimization through penalty methods with automatic dual variable adjustment, enabling more flexible policy updates while maintaining asymptotic convergence to constraint-satisfying policies. Projection-based methods (Tessler et al. 2019, Dalal et al. 2018) ensure safety through explicit projection operations onto feasible policy sets, while Lyapunov-based methods (Chow et al. 2018b) leverage control theory to construct stability certificates that guarantee convergence to safe policies while maintaining constraint satisfaction throughout learning.

The safe reinforcement learning literature encompasses different levels of safety guarantees (García and Fernández 2015). Hard constraints require deterministic satisfaction at every time step, making them suitable for safety-critical applications where any violation is unacceptable. Probabilistic constraints (Chow et al. 2018a, Jansen et al. 2020) specify satisfaction with high probability, accounting for stochastic dynamics and uncertainty. Soft constraints (Achiam et al. 2017) penalize violations through cost functions while allowing occasional transgressions, appropriate when constraints represent preferences rather than challenging

requirements. The choice of formulation depends on application requirements and the criticality of constraint violations.

Despite their diversity in constraint handling mechanisms, these model-free approaches share a fundamental limitation: they require extensive interaction with the real environment to estimate both reward and cost advantage functions accurately. Each policy update necessitates fresh rollouts to compute advantages under the current policy, resulting in sample complexity that scales poorly with problem dimensionality. This limitation is particularly acute for robotic systems where data collection is expensive, time-consuming, or potentially dangerous during exploration. The challenge of reducing this sample complexity while maintaining constraint satisfaction motivates the integration of model-based methods with constrained optimization.

3.2.2 Model-Based Reinforcement Learning

Model-based RL learns predictive dynamics models to generate synthetic experience, dramatically reducing real environment interactions (Deisenroth and Rasmussen 2011, Nagabandi et al. 2018, Chua et al. 2018). Model-Based Policy Optimization (MBPO) (Janner et al. 2019) demonstrates that branched rollouts from replay states combined with short horizons can match model-free asymptotic performance while using substantially less real data. Similar gains appear in latent space approaches that plan over learned representations (Hafner et al. 2019, 2020).

Model error remains the central challenge. Prediction errors compound over rollout steps, poisoning advantage estimates, and destabilizing learning (Wang et al. 2019). Ensemble methods address this by training multiple models with different initializations, using prediction disagreement to quantify epistemic uncertainty (Lakshminarayanan et al. 2017, Osband et al. 2016). For probabilistic models, KL divergence between ensemble predictions captures both mean and variance disagreement (Chua et al. 2018, Kurutach et al. 2018). Prior work uses this signal for exploration (Pathak et al. 2017, Sekar et al. 2020) or to terminate rollouts when uncertainty grows large (Janner et al. 2019, Kidambi et al. 2020).

These approaches optimize reward without safety constraints. Safe model-based methods exist, utilizing Gaussian processes with stability guarantees (Berkenkamp et al. 2017) or model predictive control with safety layers (Koller et al. 2018, Wachi and Sui 2020). However, these methods require restrictive assumptions about known safe regions or linear dynamics. The question of how to systematically integrate ensemble uncertainty with constrained policy optimization for general high-dimensional tasks remains open.

CMBPO provides this integration by treating ensemble disagreement as an explicit constraint within the policy optimization framework. Rather than applying fixed heuristics for model usage, the approach adaptively adjusts reliance on synthetic

data based on measured uncertainty, enabling aggressive model exploitation while maintaining approximate safety guarantees through principled data mixing and rollout control.

3.3 Background and Problem Formulation

This section establishes the theoretical foundations and formalizes the problem that CMBPO addresses. We build upon the constrained reinforcement learning formulation presented in Chapter 2 and extend it to incorporate model-based policy optimization with uncertainty quantification. The key challenge lies in reconciling the sample efficiency benefits of learned dynamics models with the need for accurate constraint satisfaction estimates.

3.3.1 Constrained Markov Decision Process

As established in Section 2.2.2, the CMDP framework extends standard MDPs with cost functions and safety constraints. We briefly recall the key elements relevant to this chapter. A CMDP augments the standard tuple with cost functions $\mathcal{C} = \{c_j\}$ and thresholds $\mathcal{D} = \{d_j\}$. The constrained optimization problem seeks policies that maximize expected return while respecting cost limits:

$$\max_{\pi} \quad J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \quad (3.1)$$

$$\text{subject to} \quad J_{c_j}(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t c_{j,t} \right] \leq d_j, \quad \forall j \quad (3.2)$$

This chapter develops methods to solve such constrained problems using learned models while explicitly managing the model uncertainty that threatens constraint satisfaction.

3.3.2 Trust Region Policy Optimization

Policy gradient methods face a fundamental challenge: we want to maximize $J(\pi')$ but can only sample trajectories under the current policy π . Direct optimization carries significant risks of destabilizing updates. Trust Region Policy Optimization (TRPO) (Schulman et al. 2015) addresses this through a surrogate objective that approximates the true improvement while remaining tractable to estimate from on-policy data:

$$L_{\pi}(\pi') = \mathbb{E}_{s \sim d^{\pi}, a \sim \pi} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^{\pi}(s, a) \right], \quad (3.3)$$

where d^π is the discounted state distribution under π and A^π is the advantage function. The importance ratio $\pi'(a|s)/\pi(a|s)$ reweights samples from π to approximate expectations under π' .

However, this surrogate only approximates true performance when the policies remain closely aligned. Large policy changes cause the importance weights to become unreliable and the state distributions to diverge. TRPO addresses this by constraining the average D_{KL} between policies:

$$\bar{D}_{\text{KL}}(\pi, \pi') = \mathbb{E}_{s \sim d^\pi} [D_{\text{KL}}(\pi'(\cdot|s) \parallel \pi(\cdot|s))] \leq \delta. \quad (3.4)$$

This constraint has theoretical justification. Kakade and Langford’s monotonic improvement bound (Kakade and Langford 2002) shows that when policy divergence remains small, improving the surrogate guarantees an improvement in true performance. In practice, TRPO solves a linearized version of this constrained optimization using conjugate gradients combined with a backtracking line search.

3.3.3 Constrained Policy Optimization

TRPO optimizes policies without considering safety. For CMDPs, we need policies that maximize return while respecting cost constraints. CPO (Achiam et al. 2017) extends the trust region framework to handle this by treating costs analogously to returns, using a surrogate to predict constraint satisfaction.

For a single cost constraint, the optimization problem becomes:

$$\begin{aligned} \max_{\pi'} \quad & \mathbb{E}_{s \sim d^\pi, a \sim \pi} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^\pi(s, a) \right] \\ \text{s.t.} \quad & \bar{D}_{\text{KL}}(\pi, \pi') \leq \delta, \\ & J_c(\pi) + \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^\pi, a \sim \pi} \left[\frac{\pi'(a|s)}{\pi(a|s)} A_c^\pi(s, a) \right] \leq d_c. \end{aligned} \quad (3.5)$$

The cost constraint mirrors the reward surrogate: starting from the current cost $J_c(\pi)$, we add a predicted cost change estimated via importance sampling using the cost advantage function A_c . This combined quantity must remain below the threshold d_c . The KL constraint serves double duty, ensuring both reward and cost surrogates remain accurate approximations.

In practice, CPO solves this constrained problem by linearizing the objective and cost constraint while using a second-order approximation for the KL divergence. The resulting convex subproblem admits a closed-form dual solution, which yields a feasible update direction. This direction is implemented via conjugate gradients with a backtracking line search to verify that both constraints are satisfied empirically.

3.3.4 Model-Based Policy Optimization

TRPO and CPO require numerous real environment interactions to accurately estimate advantages, making them prohibitively sample-inefficient for robotic applications. MBPO (Janner et al. 2019) addresses this by learning a dynamics model P_m and using it to generate synthetic rollouts for advantage estimation. However, model errors create a fundamental tension: longer rollouts provide more training data but accumulate larger prediction errors.

MBPO manages this trade-off through short, branched rollouts. Rather than generating long trajectories from a single initial state, rollouts of length k (typically 1-5 steps) start from states sampled from the replay buffer $\mathcal{D}$. This strategy covers many local regions of the state space while limiting error accumulation within each trajectory fragment. The trust region surrogate adapts naturally to model-generated data:

$$L_m^\pi(\pi') = \mathbb{E}_{s \sim d_m^\pi, a \sim \pi} \left[\frac{\pi'(a | s)}{\pi(a | s)} A_m^\pi(s, a) \right], \quad (3.6)$$

where the advantage A_m^π is computed from model rollouts and d_m^π denotes the state distribution induced by the learned model. When rollouts remain short, this surrogate closely approximates the true objective despite model imperfections.

The performance guarantee degrades gracefully with rollout length:

$$J(\pi') \approx L_m^\pi(\pi') \pm \epsilon_{\text{model}}(k), \quad (3.7)$$

where the model-induced error $\epsilon_{\text{model}}(k)$ grows with horizon k . MBPO achieves model-free asymptotic performance with dramatically fewer real samples by keeping k small enough that this error remains tolerable.

However, MBPO optimizes unconstrained objectives. The cost surrogates required for CPO face the same model error problem, but constraint violations have different consequences than suboptimal rewards. A model that underestimates costs can produce a policy that appears safe in simulation but violates constraints during real deployment. This asymmetry between optimistic reward estimates (acceptable) and optimistic cost estimates (dangerous) motivates explicit uncertainty quantification.

3.3.5 Ensemble-Based Uncertainty Quantification

The preceding discussion reveals a critical challenge: MBPO can generate policies that appear safe according to model-based cost estimates but violate constraints in reality. To address this, we need a mechanism to detect when the model produces unreliable cost predictions. Ensemble methods provide this capability by training multiple models and measuring the disagreement among them.

An ensemble consists of M dynamics models, each initialized randomly and trained on different shuffles of the replay buffer $\mathcal{D}$. Where training data is abundant and consistent, models converge to similar predictions. Where data is sparse or noisy, models diverge, with disagreement revealing epistemic uncertainty about the true dynamics (Lakshminarayanan et al. 2017, Osband et al. 2016). Figure 3.1 illustrates this behavior.

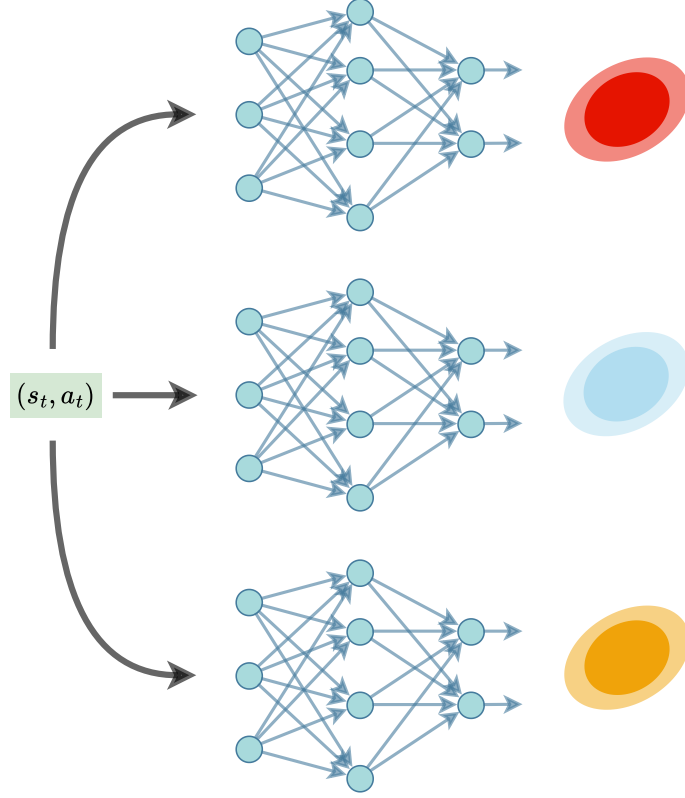


Figure 3.1: Ensemble of probabilistic dynamics models. Each model predicts a Gaussian next-state distribution. Individual model variance captures aleatoric uncertainty, while disagreement across the ensemble captures epistemic uncertainty and highlights regions where the model is undertrained (Chua et al. 2018).

Each ensemble member i models transitions as Gaussian distributions:

$$P_{\vartheta_i}(s' | s, a) = \mathcal{N}(\mu_{\vartheta_i}(s, a), \Sigma_{\vartheta_i}(s, a)), \quad \Sigma_{\vartheta_i}(s, a) = \text{diag}(\sigma_{\vartheta_i}^2(s, a)), \quad (3.8)$$

where μ_{ϑ_i} predicts the mean next state and the diagonal covariance Σ_{ϑ_i} captures process noise. Parameters are learned by minimizing the negative log-likelihood:

$$\min_{\vartheta_i} \mathbb{E}_{(s, a, s') \sim \mathcal{D}} \left[\frac{1}{2} \left\| \Sigma_{\vartheta_i}^{-\frac{1}{2}}(s, a) (s' - \mu_{\vartheta_i}(s, a)) \right\|_2^2 + \frac{1}{2} \log |\Sigma_{\vartheta_i}(s, a)| \right]. \quad (3.9)$$

This formulation separates two types of uncertainty. Aleatoric uncertainty (within-model variance Σ_{θ_i}) reflects irreducible randomness in the environment. Epistemic uncertainty (between-model disagreement) indicates insufficient training data. Modeling aleatoric uncertainty explicitly allows the ensemble to isolate epistemic effects: in stochastic but well-covered regions, models agree on the noise structure despite high individual variances, yielding low disagreement. Low disagreement thus signals reliable cost predictions even in noisy environments, while high disagreement reveals fundamental uncertainty about the dynamics, rendering cost surrogates unreliable. This asymmetry motivates ensemble disagreement as the primary signal for controlling model usage in the constrained optimization framework.

3.3.6 Problem Formulation

CMBPO integrates the components established above. CPO’s constrained framework provides the optimization structure, MBPO’s model-generated rollouts enable sample efficiency, and ensemble disagreement reveals when predictions become unreliable. The challenge lies in extending CPO’s trust region optimization to leverage these synthetic advantages while preventing model uncertainty from undermining constraint satisfaction. CMBPO addresses this by explicitly incorporating ensemble disagreement into the optimization, allowing the algorithm to balance the benefits of model-based learning against the risk of unsafe updates when predictions are uncertain.

3.4 Method

This section presents CMBPO in three parts. We first derive a performance bound that quantifies how model error affects policy improvement under safety constraints. We then introduce practical mechanisms to quantify and constrain model uncertainty using ensemble disagreement. Finally, we demonstrate how these components are integrated into a comprehensive algorithm that balances sample efficiency with constraint satisfaction.

3.4.1 Performance Bounds Under Model Dynamics

To clarify the effect of model error on constrained policy optimization, we relate the true return change to a model-based objective plus a penalty that depends on both policy divergence and model mismatch (cf. (Kakade and Langford 2002, Schulman et al. 2015, Achiam et al. 2017)). This motivates explicit uncertainty management in constrained settings.

Theorem 3.1 (Return Difference under Model Uncertainty). Let $J(\pi)$ and $J(\pi')$ denote the expected returns under true dynamics, and let d^π and d_m^π be the discounted stationary distributions under the true and learned model dynamics P_m , respectively. Then the true return difference satisfies

$$\Delta J(\pi, \pi') \in \left[\frac{L_m^\pi(\pi')}{1-\gamma} - \frac{4\delta_{\max}}{1-\gamma} \epsilon, \quad \frac{L_m^\pi(\pi')}{1-\gamma} + \frac{4\delta_{\max}}{1-\gamma} \epsilon \right], \quad (3.10)$$

where the relative performance term is

$$L_m^\pi(\pi') = \mathbb{E}_{s \sim d_m^\pi, a \sim \pi} \left[\frac{\pi'(a|s)}{\pi(a|s)} A_m^\pi(s, a) \right], \quad (3.11)$$

the tightness factor is

$$\delta_{\max} = \max_{s, a, s'} |r(s, a, s') + V_m^\pi(s') - V_m^\pi(s)|, \quad (3.12)$$

and the aggregate penalty is

$$\epsilon = \epsilon_\pi \epsilon_m + \frac{\gamma}{1-\gamma} (\epsilon_\pi^2 + 2\epsilon_\pi \epsilon_m),$$

where

$$\epsilon_\pi = \max_s D_{\text{TV}}(\pi'(\cdot|s) \parallel \pi(\cdot|s)), \quad \epsilon_m = \max_{s, a} D_{\text{TV}}(P(\cdot|s, a) \parallel P_m(\cdot|s, a)).$$

Intuitively, $L_m^\pi(\pi')$ is what we can optimize with the model, while the aggregate error term ϵ (scaled by the tightness factor $\delta_{\max}$) captures how far the model-policy pair drifts from the true dynamics. Here ϵ_π measures policy divergence and ϵ_m quantifies model error. The full derivation can be found in Appendix A.1 or in the arXiv preprint version of our paper (Zanger et al. 2021). The return bound establishes that model error creates uncertainty in performance predictions. However, for constrained problems, the critical issue is constraint satisfaction: while underestimating returns merely yields suboptimal policies, underestimating costs can produce unsafe behavior. This asymmetry motivates deriving an analogous bound for cost returns.

Implications for Constrained Optimization

An analogous bound holds for expected cost returns using the cost objective $J_c(\pi)$ and cost threshold d_c :

$$\Delta J_c(\pi, \pi') \in \left[\frac{L_{m,c}^\pi(\pi')}{1-\gamma} - \frac{4\delta_{\max,c}}{1-\gamma} \epsilon, \quad \frac{L_{m,c}^\pi(\pi')}{1-\gamma} + \frac{4\delta_{\max,c}}{1-\gamma} \epsilon \right], \quad (3.13)$$

where $L_{m,c}^\pi$ and $\delta_{\max,c}$ are defined analogously for the cost function. To guarantee $J_c(\pi') \leq d_c$, it is sufficient that

$$J_c(\pi) + \frac{L_{m,c}^\pi(\pi')}{1-\gamma} + \frac{4\delta_{\max,c}}{1-\gamma} \epsilon \leq d_c. \quad (3.14)$$

The bound reveals that safe policy improvement requires controlling both policy divergence (ϵ_π) and model error (ϵ_m). However, directly constraining the maximum total variation distances is intractable. We therefore translate these requirements into three practical constraints: a trust region constraint that bounds average policy divergence using D_{KL} (a tractable proxy for ϵ_π).

3.4.2 Three-Constraint Optimization Formulation

Formally, the optimization problem becomes:

$$\begin{aligned}
 \max_{\pi'} \quad & \mathbb{E}_{s \sim d_m^\pi, a \sim \pi} \left[\frac{\pi'(a|s)}{\pi(a|s)} A_m^\pi(s, a) \right] \\
 \text{s.t.} \quad & \mathbb{E}_{s \sim d_m^\pi} [D_{\text{KL}}(\pi'(\cdot|s) \parallel \pi(\cdot|s))] \leq d_\pi \quad (\text{trust region}) \\
 & J_c(\pi) + \frac{L_{m,c}^\pi(\pi')}{1 - \gamma} \leq d_c \quad (\text{safety}) \\
 & \mathbb{E}_{s \sim d_m^\pi, a \sim \pi} [D_{\text{KL}}(P(\cdot|s, a) \parallel P_m(\cdot|s, a))] \leq d_m \quad (\text{model uncertainty}).
 \end{aligned} \tag{3.15}$$

CMBPO solves this problem through a hybrid approach. The trust region and safety constraints are handled by CPO’s analytical optimization framework (Sec . 3.4.2), which computes closed-form policy updates. The model uncertainty constraint requires a different strategy: rather than incorporating it into the policy gradient computation, we enforce it by controlling the data generation process through adaptive mixing of real and synthetic samples and dynamically limiting rollout horizons based on ensemble disagreement (Sec . 3.4.3). We detail each component below.

Policy Updates via CPO

CPO (Achiam et al. 2017) provides an analytical solution to the constrained optimization problem in Eq. (3.15). The approach linearizes the reward and cost objectives around the current policy while imposing a quadratic trust region on the average D_{KL} .

Let θ denote the current policy parameters and θ' a proposal. The policy gradient g and cost gradient b are defined as $g = \nabla_\theta L_m^\pi(\pi_\theta)$ and $b = \nabla_\theta L_{m,c}^\pi(\pi_\theta)$, while $\mathbf{H}$ denotes the Fisher information matrix (the KL Hessian). For sufficiently small policy updates, the objectives and KL constraint admit local approximations:

$$L_m^\pi(\pi_{\theta'}) \approx g^\top (\theta' - \theta), \tag{3.16}$$

$$L_{m,c}^\pi(\pi_{\theta'}) \approx b^\top (\theta' - \theta), \tag{3.17}$$

$$\bar{D}_{\text{KL}}(\pi_\theta \parallel \pi_{\theta'}) \approx \frac{1}{2} (\theta' - \theta)^\top \mathbf{H} (\theta' - \theta). \tag{3.18}$$

These approximations reduce the original problem to a convex quadratic program. Let $c_0 := J_c(\pi_\theta) - d_c$ represent the current cost surplus relative to the threshold. The local optimization step becomes:

$$\begin{aligned} \max_{\theta'} \quad & g^\top(\theta' - \theta) \\ \text{s.t.} \quad & c_0 + \frac{1}{1-\gamma} b^\top(\theta' - \theta) \leq 0, \\ & \frac{1}{2}(\theta' - \theta)^\top H(\theta' - \theta) \leq d_\pi. \end{aligned} \quad (3.19)$$

This convex subproblem admits a closed-form solution via Lagrangian duality. Defining $\tilde{b} := \frac{b}{1-\gamma}$ and solving for the optimal Lagrange multipliers (λ^*, ν^*) , the primal update is:

$$\theta^* = \theta + \frac{1}{\lambda^*} H^{-1}(g - \nu^* \tilde{b}). \quad (3.20)$$

In practice, $H^{-1}(\cdot)$ is computed efficiently using conjugate gradients, and a backtracking line search verifies that both the empirical KL and cost constraints are satisfied. A compact derivation of the dual solution is provided in Appendix A.2.

Having established how CPO handles the trust region and safety constraints analytically, we now address the third constraint: managing model uncertainty to prevent unsafe policy updates when predictions become unreliable.

3.4.3 Ensemble-Based Uncertainty Quantification

The uncertainty constraint in Eq. (3.15) requires a practical, state-action measure of model error. Following Sec. 3.3.5, we quantify epistemic uncertainty through disagreement across the M independently trained dynamics models.

Quantifying Epistemic Uncertainty

To enforce the model uncertainty constraint in practice, we need a practical measure of epistemic uncertainty that can be computed without ground-truth data. We quantify disagreement at each state-action pair (s, a) through the expected pairwise D_{KL} among the M ensemble predictors:

$$D_{\text{KL}}^E(s, a) = \frac{1}{M(M-1)} \sum_{i \neq j} D_{\text{KL}}(P_{\vartheta_i}(\cdot | s, a) \| P_{\vartheta_j}(\cdot | s, a)). \quad (3.21)$$

This metric measures the extent to which the models disagree about the transition distribution. For the Gaussian predictors defined in Sec. 3.3.5, each pairwise divergence admits a closed form:

$$D_{\text{KL}}(P_i \| P_j) = \frac{1}{2} \left[\ln \frac{|\Sigma_j|}{|\Sigma_i|} + \text{tr}(\Sigma_j^{-1} \Sigma_i) + (\mu_j - \mu_i)^\top \Sigma_j^{-1} (\mu_j - \mu_i) - d_s \right], \quad (3.22)$$

3 Constrained Model-Based Policy Optimization

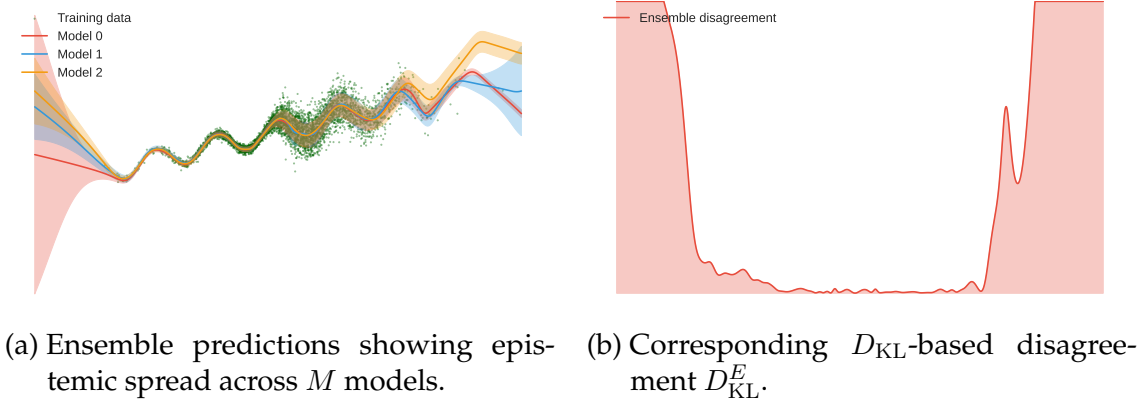


Figure 3.2: Visualization of ensemble-based uncertainty quantification. Panel (a) shows predictions from three ensemble models (colored lines with uncertainty bands) fitted to training data (green points). The models exhibit close agreement near the training distribution but diverge substantially in extrapolated regions. Panel (b) shows the corresponding ensemble disagreement D_{KL}^E computed via Eq. 3.21, which peaks in regions far from training data where the ensemble members make conflicting predictions. This disagreement metric guides adaptive mixing (Eq. 3.25) and horizon selection (Eq. 3.26).

where d_s is the state dimension. The expected disagreement under the current policy is then:

$$\bar{D}_{\text{KL}}^E(\pi) = \mathbb{E}_{s \sim d_m^\pi, a \sim \pi}[D_{\text{KL}}^E(s, a)]. \quad (3.23)$$

This disagreement metric has several practical advantages: it is invariant to state reparameterizations (units and scaling), requires no ground-truth validation at test time, and empirical studies have shown it tracks true model error effectively (Lakshminarayanan et al. 2017, Chua et al. 2018, Janner et al. 2019). Figure 3.2 illustrates this concept through a one-dimensional example. Panel (a) shows how three ensemble models trained on the same data produce increasingly divergent predictions as they extrapolate beyond the training distribution. Panel (b) quantifies this divergence through D_{KL}^E , revealing high disagreement precisely where the models are uncertain about the dynamics. With this uncertainty measure in hand, we can now enforce the model uncertainty constraint through two complementary mechanisms: adaptive data mixing and state-dependent rollout horizons.

3.4.4 Adaptive Real-Model Data Mixing

The first mechanism controls the fraction of model-generated data used in each policy update. When ensemble disagreement is high, we increase the proportion of real samples from the replay buffer $\mathcal{D}$; when predictions are consistent, we allow more synthetic data. This directly enforces the uncertainty constraint from Eq. (3.15).

Let $\alpha(\pi)$ denote the fraction of real transitions in each training batch. We set α such that the model contribution keeps expected disagreement below the threshold d_m :

$$(1 - \alpha) \bar{D}_{\text{KL}}^E(\pi) \leq d_m. \quad (3.24)$$

Solving for α yields the adaptive mixing rule:

$$\alpha(\pi) = \max\left(0, 1 - \frac{d_m}{\bar{D}_{\text{KL}}^E(\pi)}\right). \quad (3.25)$$

This formulation has an intuitive interpretation: when $\bar{D}_{\text{KL}}^E(\pi) > d_m$, the mixing fraction α increases proportionally to reduce reliance on uncertain model predictions. When $\bar{D}_{\text{KL}}^E(\pi) \leq d_m$, we can safely use purely model-generated data ($\alpha = 0$). In practice, α typically decreases over training as the model improves and disagreement drops.

3.4.5 Uncertainty-Capped Rollout Horizons

The second mechanism controls how long each model rollout continues before termination. Standard MBPO uses fixed rollout lengths with a time-based schedule that gradually increases the horizon k over training (Janner et al. 2019). This global approach treats all regions of state space as equally reliable.

CMBPO instead treats model accuracy as spatially heterogeneous: some regions have abundant training data and low disagreement, while others remain uncertain. We therefore cap each rollout based on its accumulated disagreement rather than a global step count. Starting from an initial state s_0 sampled from the replay buffer, a model trajectory continues until:

$$\sum_{\tau=0}^{h-1} D_{\text{KL}}^E(s_\tau, a_\tau) \leq d_H, \quad (3.26)$$

where h is the realized rollout length and d_H is the cumulative disagreement budget. This yields longer rollouts in well-modeled regions where the ensemble is in agreement, and terminates early in uncertain regions where predictions diverge. The approach naturally adapts to local model quality without requiring manual horizon schedules.

Together, adaptive mixing and uncertainty-capped horizons provide complementary controls on model usage: mixing adjusts the global balance between real and synthetic data. In contrast, horizon capping adjusts the length of individual trajectories based on local reliability.

3.5 Experimental Validation

This section evaluates CMBPO on constrained continuous-control tasks to answer four central questions:

1. Does CMBPO maintain safety constraints throughout training?
2. Does model usage actually improve sample efficiency?
3. How does final performance compare to strong baselines?
4. What role do adaptive resampling and uncertainty-capped rollouts play in achieving these results?

We address these questions through experiments on simulated robot locomotion tasks that create genuine conflicts between reward maximization and constraint satisfaction.

3.5.1 Experimental Setup

Our experiments use the MuJoCo physics simulator (Todorov et al. 2012) to evaluate CMBPO on high-dimensional continuous control problems. We selected three locomotion tasks that represent fundamentally different safety challenges: static obstacles requiring spatial awareness, dynamic hazards demanding temporal prediction, and path-following constraints testing precise control under uncertainty.

Evaluation Environments

The **AntSafe** environment (Fig. 3.3a) presents a spatial reasoning challenge. A quadruped robot must learn to navigate forward locomotion while avoiding overturnable walls. The agent earns rewards for forward velocity but incurs costs when approaching walls too closely or falling over. This creates a natural tension: moving quickly increases returns but raises collision risk, forcing the policy to balance speed against spatial safety margins.

The **HalfCheetahSafe** task (Fig. 3.3b) introduces temporal uncertainty through a dynamic obstacle. The planar robot must maintain forward speed while avoiding a moving object ahead. Unlike static obstacles, this requires the agent to predict future states and adjust behavior preemptively. Costs accumulate when the robot gets too close to the obstacle, demanding policies that learn when to slow down or navigate around hazards based on their predicted trajectories.

AntCircle (Fig. 3.3d) tests precise control under continuous constraints (Achiam et al. 2017). The quadruped must run along a circular path while staying within a designated corridor. The reward-maximizing trajectory lies along the green circle; entering the blue outer regions violates the constraints. Success requires balancing

forward momentum with accurate path following, making this task particularly sensitive to the quality of cost predictions.

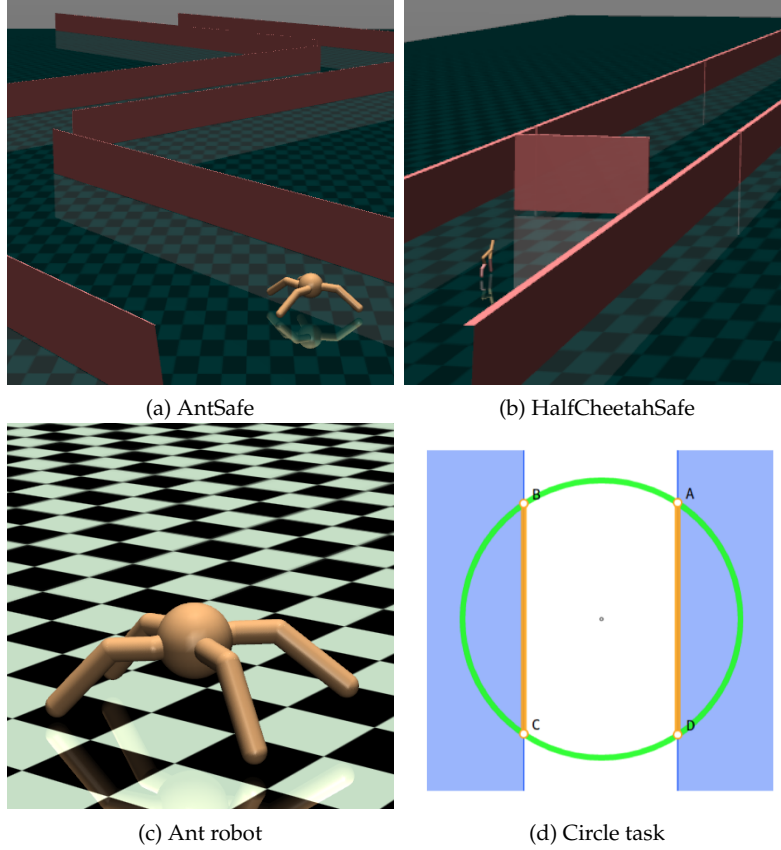


Figure 3.3: Evaluation environments. (a) AntSafe rewards forward locomotion while requiring the ant to avoid overturnable walls. (b) HalfCheetahSafe rewards forward locomotion while requiring the robot to maintain a safe distance from a moving obstacle. (c) Ant robot morphology used for both AntSafe and AntCircle tasks. (d) AntCircle task, where the ant must follow a circular track while avoiding blue violation zones (Achiam et al. 2017). Images reproduced from Zanger et al. (2021). © 2021 IEEE.

Both robots receive proprioceptive observations, including joint positions, orientations, velocities, and global center of mass location. Control occurs through continuous torque commands at each timestep. These high-dimensional state and action spaces create substantial challenges for both model learning and safe exploration.

Baseline Algorithms

To isolate the contributions of model-based learning and uncertainty management, we compare CMBPO against representative model-free safe RL methods. CPO

(Achiam et al. 2017) provides the closest comparison, using the same trust region framework with linearized cost constraints but requiring real environment samples for all advantage estimates. Trust Region Policy Optimization with Lagrangian Methods (TRPOLag) and Proximal Policy Optimization with Lagrangian (PPO-Lag) represent the Lagrangian relaxation approach, converting the constrained problem into an unconstrained dual formulation (Ray et al. 2019). We also include unconstrained TRPO (Schulman et al. 2015) to demonstrate the consequences of ignoring safety entirely.

These baselines establish whether CMBPO’s sample efficiency gains come at the cost of constraint satisfaction quality. If model-based learning merely trades safety for speed, we would expect CMBPO to reach high returns quickly while violating constraints. The experiments test whether adaptive uncertainty management prevents this failure mode.

All methods use identical network architectures and training hyperparameters where applicable. Complete settings, including CMBPO-specific parameters such as ensemble size (M equals 7), elite selection (5 models), and initial mixing ratios (0.2 to 0.4), are provided in Appendix A.3.

Metrics and Reporting

We track five key metrics throughout training. Mean episodic return measures the average cumulative reward per episode, indicating task performance. Mean episodic cost tracks safety by measuring average cumulative constraint violations, which should stay below the limit d_c . Sample efficiency counts how many real environment interactions are needed to reach target performance levels. Cumulative training cost sums all constraint violations from the start of training until a safe policy emerges, capturing the total safety cost of learning. Finally, constraint satisfaction rate measures what fraction of training time the current policy actually satisfies the safety constraint.

We run 4 independent trials with different random seeds for CMBPO and 2 trials for baseline algorithms due to computational constraints. All plots show mean values across these seeds with shaded regions indicating plus or minus one standard deviation. We use logarithmic scaling on the horizontal axis to emphasize differences in sample efficiency, plotting against total real environment interactions. A dashed horizontal line marks the cost constraint threshold for easy visual reference.

3.5.2 Main Results

We present training performance across all algorithms and tasks, followed by cumulative cost analysis and ablation studies.

Training Performance

Figure 3.4 shows training curves for all five algorithms across the three evaluation tasks. Each plot tracks mean episodic return (top) and mean episodic cost (bottom) against total real environment interactions on a logarithmic scale.

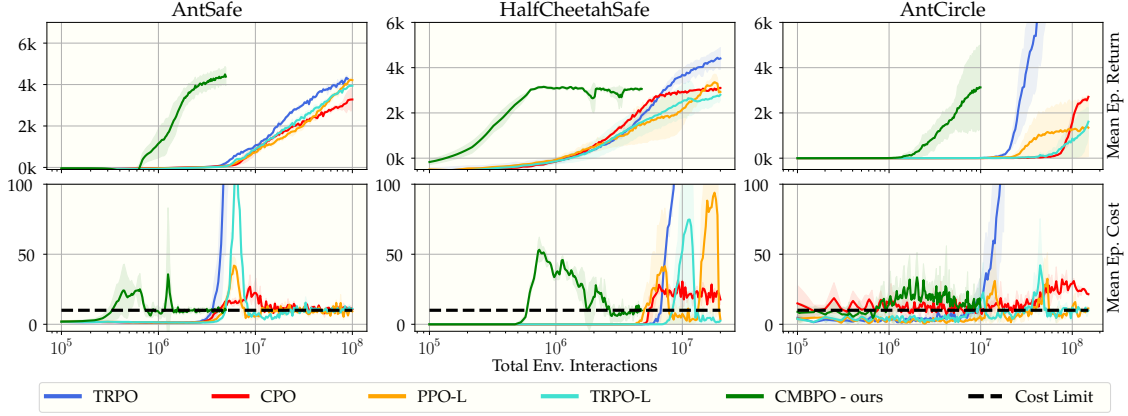


Figure 3.4: Training performance across all algorithms and environments. Solid lines show mean episodic return and cost over 4 random seeds for CMBPO and 2 random seeds for the baseline algorithms, and shaded bands indicate one standard deviation. CMBPO achieves higher returns at comparable or lower mean episodic cost than model-free baselines across AntSafe, AntCircle, and HalfCheetahSafe. The dashed black line marks the cost constraint limit d_c . Adapted from Zanger et al. (2021). © 2021 IEEE.

Sample efficiency differences appear consistently across all tasks. CMBPO reaches a return of 2500 on AntSafe using approximately 200,000 real environment samples, whereas CPO, TRPOLag, and PPO-Lag require roughly 2 million samples, a tenfold reduction. Similar patterns emerge on HalfCheetahSafe, where CMBPO achieves a return of 3000 with 300,000 samples compared to 2-3 million for model-free methods (7-10 $\times$ improvement). AntCircle shows the most dramatic gap: CMBPO needs 1 million samples to reach a return of 2000, while baselines require 10-20 million samples.

Cost trajectories reveal different constraint adherence patterns. Unconstrained TRPO consistently exceeds the cost limit throughout training on all tasks. Among constrained algorithms, CPO, TRPOLag, and PPO-Lag maintain tighter cost control during early training. CMBPO exhibits more variability in this early phase, with costs occasionally exceeding the limit before converging below it later. Final asymptotic performance shows CMBPO achieving comparable or higher returns than model-free methods while satisfying constraints, with slightly higher returns on AntCircle.

Cumulative Cost Analysis

Mean episodic cost tracks average constraint violations per episode during training. Cumulative training cost sums all violations from training start until achieving safe performance at a specified return level. This metric quantifies the total safety cost incurred throughout the learning process.

Table 3.1 reports cumulative costs measured at the return thresholds shown for each task. Values represent total constraint violations integrated across all training episodes up to the point where the algorithm achieves both the target return and constraint satisfaction.

Environment	CMBPO	CPO	PPOLag	TRPOLag
AntSafe@2500	18	527	422	385
HalfCheetahSafe@3000	58	100	515	223
AntCircle@2000	107	2395	1548	1576

Table 3.1: Total constraint violations accumulated during training until achieving the target return level. All cost values are in thousands. Values adapted from Zanger et al. (2021). © 2021 IEEE.

Despite showing more transient violations during early training, CMBPO accumulates substantially lower total costs than all baselines. The reductions range from approximately $2\times$ on HalfCheetahSafe to over $20\times$ on AntCircle and AntSafe. These differences reflect both faster convergence (fewer training episodes overall) and the protective effect of uncertainty-aware model usage during learning.

Ablation Studies

To isolate the contribution of adaptive mechanisms, we evaluate three CMBPO variants on AntSafe and AntCircle. Each variant removes or replaces one adaptive component while keeping remaining algorithm components identical.

The first variant employs a fixed mixing ratio rather than adaptive mixing based on ensemble disagreement. The ratio is set to 5% real data on AntSafe and 3% on AntCircle, values selected via hyperparameter search to perform well under constant mixing.

The second and third variants replace uncertainty-capped rollout horizons with predetermined schedules. Schedule1 increases rollout length linearly from 3 to 15 steps over 1000 training epochs. Schedule2 uses a more aggressive schedule, increasing from 5 to 25 steps over the same period.

Figure 3.5 shows training curves for all variants alongside full CMBPO. Each curve averages results across 4 random seeds.

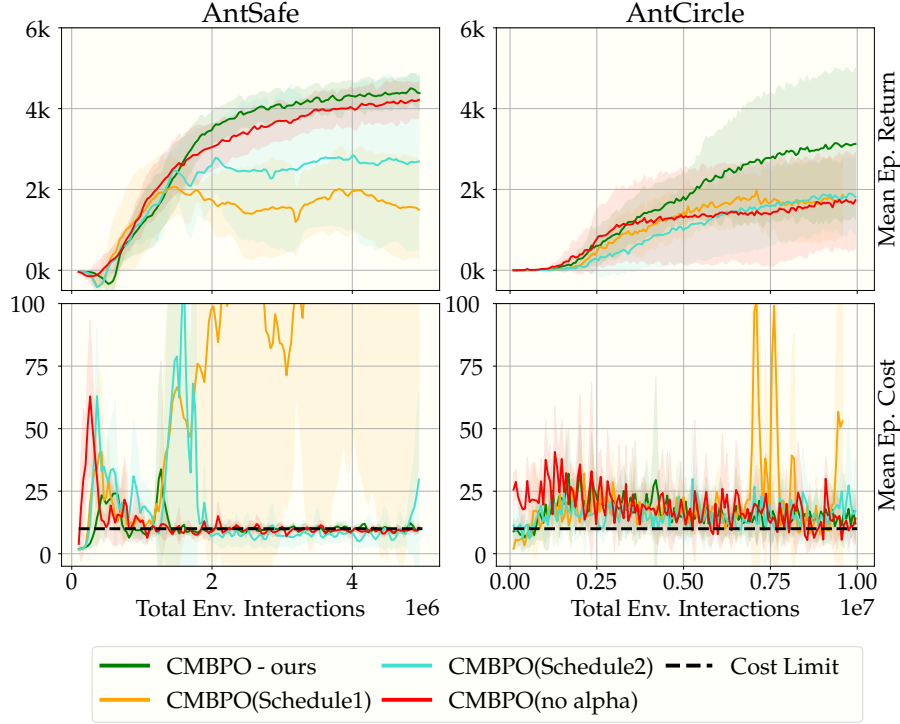


Figure 3.5: Ablation results on AntSafe and AntCircle comparing full CMBPO with variants that use fixed mixing ratios and scheduled horizon caps. Solid lines denote mean return and cost over 4 random seeds, and shaded regions show one standard deviation. The dashed line marks the cost limit d_c . Adapted from Zanger et al. (2021). © 2021 IEEE.

Both scheduled-horizon variants exhibit sudden performance drops on AntSafe that coincide with cost spikes beyond the limit. Schedule1 shows a sharp decrease around 2-3 million environment steps, with returns falling from approximately 4000 to below 2000. Schedule2 experiences similar collapses with slower recovery. The fixed-mixing variant maintains more stable curves but converges to slightly lower final returns than full CMBPO (approximately 3500 versus 4000).

AntCircle shows more pronounced instability. Both scheduled-horizon variants undergo multiple performance collapses. Schedule2 degrades particularly severely, with returns dropping below 1000 after initial progress and failing to recover within the training budget. Schedule1 exhibits similar instability with repeated oscillations. The fixed-mixing variant displays higher variance in both return and cost trajectories, with several seeds converging to substantially lower returns ranging from 1500 to 3500.

Full CMBPO maintains more consistent training progress on both tasks. Training curves show smoother convergence with fewer abrupt changes. Final returns reach 4000 on AntSafe and approximately 3000 on AntCircle, matching or exceeding all ablation variants while maintaining better constraint satisfaction during training.

3.6 Discussion and Limitations

3.6.1 Sample Efficiency vs Safety Tradeoffs

The experimental results demonstrate substantial improvements in sample efficiency while maintaining asymptotic constraint satisfaction, but with an important tradeoff during early training. CMBPO achieves 10-20 $\times$ reductions in required samples by leveraging model-generated rollouts for advantage estimation. This acceleration stems from the branched rollout scheme, where short trajectories starting from replay buffer states provide local training data without accumulating severe prediction errors. The ensemble disagreement mechanism ensures this synthetic data remains useful by restricting model reliance when predictions become unreliable.

However, early training exhibits more frequent transient violations than model-free baselines. This occurs because short rollout horizons limit the quality of advantage estimates when both model and policy are still developing. The cost advantage function requires accurate predictions over multiple steps to estimate whether a policy update will satisfy constraints. When models are young and rollouts are brief, these estimates contain more uncertainty, occasionally leading to updates that violate constraints despite appearing safe according to the model-based surrogate.

The cumulative cost analysis reveals why this tradeoff favors CMBPO in practice. Despite early violations, total accumulated costs remain dramatically lower, often 20 $\times$ less than model-free methods. Faster convergence means fewer training episodes overall, and the uncertainty constraint prevents catastrophic failures even when individual predictions are imperfect. The theoretical bounds from Section 3.4 help explain this: controlling ensemble disagreement limits the model error term ϵ_m , keeping the cost surrogate reliable enough for approximate safety despite imperfections.

3.6.2 Role of Adaptive Mechanisms

The ablation studies reveal why both adaptive mechanisms contribute meaningfully to stable training. Scheduled rollout horizons fail because they treat model accuracy as a global property that improves monotonically with training time. In reality, model quality varies dramatically across the state-action space. A schedule may lead to prolonged rollouts in regions where the model remains uncertain, causing prediction errors to compound and corrupt the training data. Simultaneously, longer rollouts require more aggressive bootstrapping from value estimates, which can destabilize learning, particularly in environments with contact dynamics, such as AntSafe, where sudden changes make value approximation challenging.

State-dependent horizon capping addresses this by adapting to local model reliability. When ensemble predictions agree in a region, the system allows extended rollouts to gather more data and reduce distribution shift. When disagreement signals uncertainty, rollouts terminate early to limit damage from accumulated errors. This local adaptation proves far more robust than any global schedule tested, preventing the sudden performance collapses observed with fixed schedules.

Adaptive mixing based on ensemble disagreement shows subtler but meaningful effects. The mechanism automatically reduces model reliance when disagreement spikes, providing principled responses to changes in model quality without requiring task-specific tuning. While fixed mixing ratios can perform reasonably well when carefully tuned, they offer no guarantees of performance. They may allow excessive model usage during periods of high uncertainty or restrict it unnecessarily when models improve.

3.6.3 When CMBPO Succeeds and Fails

CMBPO performs well on continuous control tasks with moderate stochasticity where state-action regions receive reasonable data coverage. The three locomotion environments tested represent typical scenarios for robotic learning: spatial reasoning, temporal prediction, and precise control under continuous constraints. In these settings, ensemble methods effectively capture epistemic uncertainty, and short rollouts limit error accumulation sufficiently for stable learning.

The method sometimes exceeds model-free asymptotic performance, particularly on AntCircle. Two factors likely contribute. First, temporary constraint violations during exploration may enable the discovery of high-reward regions that overly cautious policies never reach. Second, using deterministic mean predictions from models rather than sampling may reduce gradient variance, leading to more stable policy updates. While speculative, these observations suggest model-based methods may offer benefits beyond pure sample efficiency.

However, preliminary experiments on safety-gym benchmarks revealed significant limitations. CMBPO performed poorly on tasks with partial observability and strong covariances between state dimensions. The probabilistic ensemble models assume conditional independence given state-action pairs, an assumption that breaks down when hidden variables induce correlations. Additionally, function approximation struggles to capture complex multivariate dependencies, resulting in models that make systematically biased predictions, even with abundant data.

3.6.4 Limitations

The safety-gym failures represent the primary limitation of this approach. Extending CMBPO to partially observable environments would require fundamentally different model architectures, such as recurrent networks or transformers that

capture temporal dependencies. Similarly, handling strong state covariances demands models that explicitly represent multivariate structure rather than treating dimensions independently.

A second inherent limitation involves the tradeoff between the bootstrapping model error and short-horizon rollouts. Limiting horizons controls model error accumulation but increases reliance on value function estimates for credit assignment. This creates exposure to value instability, particularly in environments with discontinuous dynamics, such as contact. While uncertainty-capped horizons mitigate this issue by adapting to local conditions, they cannot entirely eliminate it.

Computational requirements present practical constraints. Training an ensemble of seven dynamics models incurs additional overhead compared to model-free methods, although this cost is amortized across the significantly reduced number of environment interactions. For applications where simulation is cheap but computation is expensive, this tradeoff may be unfavorable. Additionally, the early training violations, although acceptable given the reduced cumulative costs, may be unacceptable in domains with an extremely low tolerance for any safety violations during learning.

Finally, the approach assumes environments are approximately Markovian with moderate stochasticity. Highly stochastic environments, where aleatoric uncertainty dominates, may provide an insufficient signal for practical model learning, while deterministic environments with chaotic dynamics may exhibit compounding errors despite low ensemble disagreement.

4 Robotic Control Using Model-Based Meta-Adaptation

4.1 Overview

Meta-reinforcement learning enables agents to rapidly adapt to new tasks by leveraging experience from related environments. This capability proves particularly valuable in robotics, where systems often encounter novel conditions that require dynamic behavioral adjustments. However, optimizing purely for adaptation speed and task performance creates a fundamental problem: rapid adaptation does not inherently guarantee that the resulting behaviors will be safe, appropriate, or aligned with operational requirements.

Traditional meta-learning approaches such as Model-Agnostic Meta-Learning (MAML) (Finn et al. 2017) and REPTILE (Nichol et al. 2018) focus exclusively on minimizing adaptation time and maximizing task rewards. While producing impressive sample efficiency gains, this objective neglects crucial considerations for deployment. Consider a delivery drone trained to navigate urban environments using meta-reinforcement learning. When encountering unexpected high winds, a purely performance-driven controller may adapt by exploiting aggressive maneuvers such as steep climbs or rapid descents to maintain forward progress. From the reward function’s perspective, this succeeds: the drone reaches its target. From an operational perspective, it fails: such maneuvers may violate airspace regulations, increase collision risk, or exceed actuator limits. The controller adapts quickly yet produces behaviors that are technically optimal but practically unacceptable.

This chapter introduces Model-Based Meta-Adaptation with Behavioral Constraints (Meta Adaptation Controller (MAC)), a framework that incorporates behavioral appropriateness directly into the adaptation process. Rather than encoding behavioral preferences through manual reward engineering for each task, MAC transfers validated behaviors through learned task embeddings. The key insight is that behavioral patterns can be transferred across different tasks using learned embedding spaces, enabling behavioral specification without per-task reward engineering.

MAC builds on the Fast Adaptation through Meta-Learning Embeddings (FAMLE) framework (Kaushik et al. 2020), which uses task embeddings to enable rapid dynamics adaptation. MAC extends this by additionally leveraging the learned embedding space to specify and enforce behavioral constraints. The approach

employs dual task-conditioned dynamics models during deployment. The first model conditions on an embedding adapted for the current environment, enabling accurate prediction of state evolution under current conditions. The second model conditions on a reference task embedding corresponding to validated operational patterns. During planning, MAC constrains predicted trajectories to maintain similarity with reference behaviors while maximizing task reward, transferring preferred behaviors across tasks without requiring scenario-specific reward functions.

The chapter proceeds as follows. Section 4.2 reviews related work in meta-learning, model-based reinforcement learning, and constrained optimization. Section 4.3 establishes the necessary background and problem formulation. Section 4.4 presents the MAC algorithm in detail, including the meta-learning procedure, online adaptation mechanism, and constrained planning approach. Section 4.5 provides experimental validation across diverse robotic locomotion tasks. Section 4.6 concludes with limitations and future directions.

4.2 Related Work

The MAC framework integrates ideas from meta-learning, model-based reinforcement learning, task representation, and constrained control. This section reviews relevant prior work and positions our contributions.

4.2.1 Meta-Learning for Fast Adaptation

Meta-learning enables rapid adaptation to new tasks with minimal data by learning across distributions of related tasks (Schmidhuber 1987). The fundamental premise is that experience from related scenarios provides inductive biases that facilitate faster learning on novel tasks.

Gradient-based meta-learning approaches learn initial parameters such that few gradient steps on new tasks yield effective solutions through bi-level optimization (Finn et al. 2017). MAML performs task-specific updates in an inner loop while optimizing the initialization in an outer loop. However, second-order derivatives impose computational overhead, motivating the use of first-order approximations that update meta-parameters toward task-specific parameters without computing higher-order gradients (Nichol et al. 2018). Black-box meta-learning offers an alternative by learning adaptation procedures through recurrent networks or other architectures that process task demonstrations directly (Duan et al. 2016), avoiding explicit gradient-based fine-tuning.

Context-based meta-learning offers an alternative paradigm by explicitly representing task identity through learned embeddings. Probabilistic latent context

variables learned from task experience enable both exploration for context inference and exploitation for task performance (Rakelly et al. 2019). Extending this with variational inference over belief states maintains uncertainty estimates that guide exploration in novel environments (Zintgraf et al. 2020). These methods demonstrate that explicitly modeling task identity through learned representations improves both sample efficiency and generalization.

While meta-learning has demonstrated significant improvements in adaptation speed and sample efficiency across diverse robotic domains, this work has primarily focused on performance metrics, such as reward maximization. Limited attention has been given to safety or behavioral appropriateness during adaptation. MAC addresses this gap by explicitly incorporating behavioral constraints that preserve preferred operational patterns during rapid adaptation to new conditions.

4.2.2 Model-Based Meta-Reinforcement Learning

Model-based meta-reinforcement learning trains dynamics models that can rapidly adapt to changing environmental conditions. Rather than learning fixed models that fail when conditions shift, meta-learned models adjust their predictions based on recent observations, maintaining accurate forecasts across diverse scenarios (Nagabandi et al. 2019). This adaptive capability proves essential for robotics, where systems encounter unexpected damage, terrain variations, or external disturbances.

A fundamental challenge arises when the distribution of tasks exhibits high diversity. Learning a single set of initial parameters that enables fast adaptation across all possible conditions becomes difficult when operational modes differ substantially. The FAMLE framework addresses this through task-conditioned dynamics models with learned embeddings that capture task-specific characteristics (Kaushik et al. 2020). The model takes the form $\hat{s}_{t+1} = f_{\theta}(s_t, a_t, h_i)$ where h_i represents task T_i . By meta-training multiple initialization points encoded through embeddings, the system selects the most appropriate starting point for adaptation in new environments, significantly improving performance for robots encountering novel operational conditions.

MAC extends this architecture by introducing reference task embeddings to constrain adapted behavior. The approach maintains rapid adaptation through embedding optimization while using a fixed reference embedding to generate trajectories that reflect validated behavioral patterns. By constraining predicted trajectories under the adapted embedding to maintain similarity with trajectories generated under the reference embedding, MAC ensures that adapted behaviors remain consistent with preferred operational patterns.

4.2.3 Imitation Learning and Behavioral Specifications

Imitation learning provides methods for acquiring behaviors from demonstrations, offering alternatives to manual reward engineering. Direct supervision from expert trajectories enables policy learning but suffers from distribution mismatch between demonstrated and encountered states (Pomerleau 1991). Iterative approaches address this limitation by collecting data under the learned policy with expert corrections, thereby improving robustness to compounding errors (Ross et al. 2011).

Adversarial training formulations enable learning from demonstrations without explicit reward specification (Ho and Ermon 2016). These methods learn reward functions that distinguish expert demonstrations from policy trajectories, though they require substantial expert data and careful tuning of the training process. Inverse reinforcement learning offers a complementary perspective by recovering reward functions that explain expert behavior, providing interpretable cost structures (Abbeel and Ng 2004, Ziebart et al. 2008).

While these methods enable learning from demonstrations, they typically require extensive expert data and assume demonstrations are available for each new task. MAC introduces a more flexible approach by meta-learning task embeddings that encode the dynamics of a reference task. During adaptation to new tasks, the approach constrains predicted trajectories to maintain similarity with trajectories generated under the reference task embedding, transferring validated behavioral patterns without requiring task-specific demonstrations. This reduces data requirements while preserving behavioral appropriateness across novel scenarios.

4.2.4 Model Predictive Control with Constraints

Model Predictive Control (MPC) provides a principled framework for handling constraints, uncertainty, and multiple objectives in robotic systems (Camacho and Bordons 2013). Classical formulations optimize control inputs over finite horizons subject to dynamics and constraint specifications. Theoretical foundations establish conditions under which recursive feasibility and stability can be guaranteed, typically requiring terminal constraint sets or cost functions that ensure convergence (Mayne et al. 2000, Rawlings et al. 2017).

Integrating learned dynamics models with MPC enables adaptation without the need for extensive system identification (Williams et al. 2017). Sampling-based optimization methods such as Cross-Entropy Method (CEM) (Rubinstein 1999) and Model Predictive Path Integral Control (MPPI) (Williams et al. 2017) enable real-time planning with neural network models while naturally handling constraints. These methods prove particularly valuable when gradient-based optimization becomes intractable due to model complexity or non-differentiable objectives.

MAC introduces a distinct constraint formulation based on behavioral similarity rather than state or input limits. Rather than specifying explicit constraint sets for each scenario, the approach constrains predicted trajectories under the adapted embedding to maintain similarity with trajectories generated under a reference task embedding that corresponds to validated operational conditions.

4.3 Background and Problem Formulation

This section establishes the technical foundations for the MAC framework and formalizes the problem it addresses. We build upon the meta-learning and model-based reinforcement learning concepts introduced in Chapter 2, extending them to incorporate behavioral constraints through task-conditioned dynamics models.

4.3.1 Meta-Learning Foundation

Building on the meta-learning framework introduced in Section 2.3.2, we consider a distribution of tasks $p(\mathcal{T})$ where each task $\mathcal{T}_i$ represents an MDP. Tasks within this distribution differ in their dynamics due to variations in environmental properties such as gravitational settings, terrain characteristics, actuator degradation, or payload changes. These variations alter how actions affect state transitions, requiring the agent to adapt its predictions and control strategies accordingly.

The meta-learning objective learns initial parameters θ that enable rapid adaptation to new tasks through gradient-based fine-tuning. Given a small dataset $\mathcal{D}_i$ from task $\mathcal{T}_i$, adaptation produces task-specific parameters through K gradient steps. The meta-objective optimizes θ for expected post-adaptation performance across the task distribution. This framework provides the foundation for MAC’s approach, which extends meta-learning to explicitly preserve behavioral appropriateness during adaptation to new conditions.

4.3.2 Model-Based Meta-Learning with Task Embeddings

Standard model-based approaches learn a single dynamics model that struggles to generalize when environmental conditions change substantially. Meta-learning addresses this by training models that can rapidly adapt their predictions; however, gradient-based adaptation of millions of model parameters remains sample-intensive, even with good initializations.

Task embeddings offer a more efficient adaptation mechanism, as demonstrated by the FAMLE framework (Kaushik et al. 2020). Rather than adapting the full model parameters θ , we augment the dynamics model with a low-dimensional

task embedding vector $h \in \mathbb{R}^d$ that captures task-specific characteristics. The task-conditioned dynamics model takes the form:

$$\hat{s}_{t+1} = f_\theta(s_t, a_t, h) \quad (4.1)$$

where f_θ represents the meta-learned model parameterized by θ , and h encodes information about the current task that influences how actions affect state transitions.

During meta-training, we jointly optimize both the model parameters θ and task-specific embeddings $\{h_i\}_{i=1}^N$ for N training tasks. The objective minimizes prediction error across all tasks:

$$\min_{\theta, \{h_i\}} \sum_{i=1}^N \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}_i} [\|s_{t+1} - f_\theta(s_t, a_t, h_i)\|^2] \quad (4.2)$$

This joint optimization discovers model parameters θ that perform well across the task distribution when combined with appropriate embeddings, while simultaneously learning embedding values that capture the distinctive characteristics of each training task.

The key advantage emerges during deployment. When encountering a new task $\mathcal{T}_{\text{new}}$, we fix the meta-learned parameters θ and optimize only the embedding h_{new} using recent observations:

$$h_{\text{new}} = \arg \min_h \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}_{\text{adapt}}} [\|s_{t+1} - f_\theta(s_t, a_t, h)\|^2] \quad (4.3)$$

where $\mathcal{D}_{\text{adapt}}$ contains the limited interaction data collected online.

Optimizing over the low-dimensional embedding space (typically 64-128 dimensions) requires far fewer samples than adapting millions of model parameters. This enables rapid adaptation with minimal interaction in the real world. The learned embeddings serve dual purposes: they enable fast dynamics adaptation through compact parameterization, and they provide a structured representation space for comparing and transferring behaviors across tasks.

4.3.3 Model Predictive Control

MPC optimizes control decisions over finite horizons while respecting dynamics and constraints. At each timestep, MPC solves:

$$\begin{aligned} & \max_{a_0, \dots, a_{H-1}} \sum_{t=0}^{H-1} \gamma^t r(s_t, a_t) \\ & \text{subject to } s_{t+1} = f(s_t, a_t), \quad t = 0, \dots, H-1 \\ & \quad s_t \in \mathcal{S}, \quad a_t \in \mathcal{A}, \quad t = 0, \dots, H-1 \end{aligned} \quad (4.4)$$

where $\mathcal{S}$ and $\mathcal{A}$ define the feasible state and action regions.

The receding horizon principle executes only the first action from the optimized sequence, then re-solves the optimization problem at the next timestep with updated state information. This approach provides robustness to model errors and disturbances by continuously incorporating new observations rather than committing to a full trajectory planned in advance.

MAC extends this framework by employing MPC with task-conditioned dynamics models $f_\theta(s_t, a_t, h)$. The constrained optimization problem is solved using the CEM, described next.

Cross-Entropy Method for Action Optimization

MAC employs the CEM to solve the constrained optimization problem in MPC. CEM is a sampling-based optimization method particularly effective for high-dimensional continuous action spaces where gradient-based methods prove difficult due to constraints or non-differentiable objectives (Rubinstein 1999).

The method maintains a parametric distribution $\mathcal{N}(\mu, \Sigma)$ over action sequences and iteratively refines it toward high-quality solutions. At each iteration, CEM samples a population of candidate action sequences from the current distribution, evaluates them according to the objective and constraints, selects an elite subset of the best-performing samples, then updates the distribution parameters to fit these elites. This process concentrates the distribution around promising regions of the action space.

For action sequences $\mathbf{a} = (a_0, \dots, a_{H-1})$ over planning horizon H , CEM iteratively updates:

$$\mu_{k+1}, \Sigma_{k+1} \leftarrow \text{fit}(\{\mathbf{a}^{(j)} \mid \mathbf{a}^{(j)} \in \text{Elite}_k\}) \quad (4.5)$$

where Elite_k denotes the top-performing samples at iteration k . The procedure terminates after a fixed number of iterations, returning the action sequence with highest objective value from the final elite set.

4.3.4 Problem Formulation

Standard meta-learning optimizes for rapid adaptation and task performance by maximizing expected return. However, achieving high rewards does not guarantee behaviorally appropriate responses. When a quadruped encounters reduced gravity, for instance, the adapted policy may discover that jumping or rolling achieves higher velocity than walking, successfully maximizing reward while producing operationally unacceptable behavior. Manually designing reward functions that produce appropriate behavior across all environmental variations proves extremely difficult, as different conditions require different implicit constraints. The challenge lies in preserving behavioral appropriateness during rapid

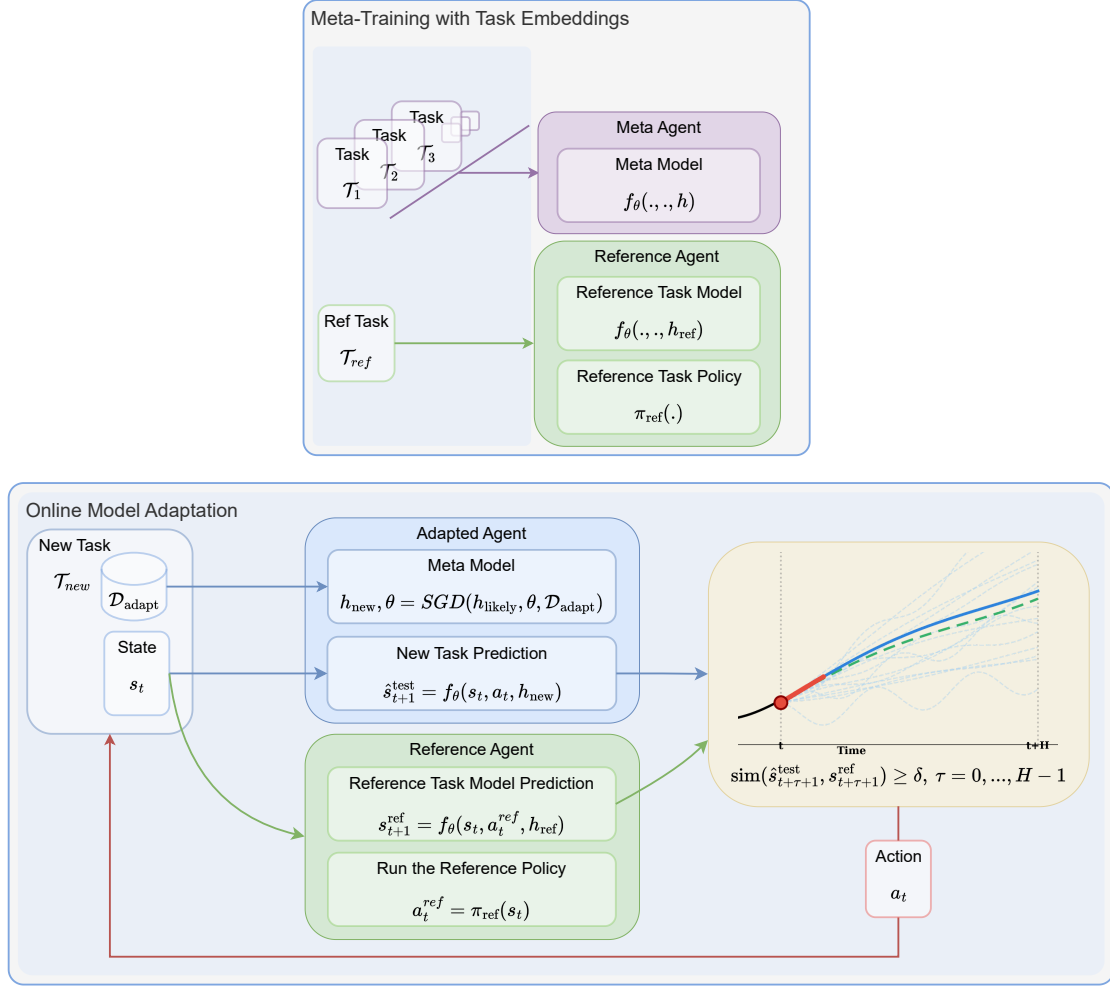


Figure 4.1: Overview of the MAC architecture divided into two phases. The **top panel** illustrates *Meta-Training with Task Embeddings*, where a Meta Agent and a Reference Agent are trained across a distribution of tasks ($\mathcal{T}_{1..3}$) and a reference task ($\mathcal{T}_{ref}$) to learn a shared model initialization. The **bottom panel** depicts *Online Model Adaptation* on a new task ($\mathcal{T}_{new}$). First, the Meta Model is updated via SGD using adaptation data $\mathcal{D}_{adapt}$ to generate task-specific parameters (h_{new}). Then, the system predicts the test trajectory ($\hat{s}^{test}$) and compares it to a reference trajectory (s^{ref}) generated by the fixed Reference Policy π_{ref} . An action a_t is executed subject to a trajectory similarity constraint ($\text{sim} \geq \delta$) over a horizon H , ensuring the adapted behavior remains close to the safe reference dynamics.

adaptation without requiring task-specific reward engineering. MAC addresses this by leveraging reference task embeddings to constrain adapted behavior. The approach uses the learned embedding space to maintain consistency with validated operational patterns while preserving the sample efficiency of model-based meta-learning.

4.4 Method

This section presents MAC in four parts. We first describe the meta-training procedure that jointly learns task-conditioned dynamics models and task embeddings from a distribution of training environments. We then introduce the dual embedding mechanism that enables behavioral similarity constraints by comparing predicted trajectories under different environmental conditions. Third, we detail the online adaptation procedure that refines both model parameters and task embeddings using recent observations. Finally, we present the constrained planning approach that employs model predictive control to select actions maximizing task reward while maintaining behavioral consistency with the reference policy. Figure 4.1 provides an overview of the complete MAC architecture.

4.4.1 Meta-Training with Task Embeddings

The meta-training phase learns both the dynamics model parameters θ and task-specific embeddings across a distribution of training tasks. We sample N distinct tasks from $p(\mathcal{T})$, where each task $\mathcal{T}_i$ represents different environmental conditions such as varying gravity, terrain properties, or actuator characteristics. For each sampled task, we collect interaction data $\mathcal{D}_i = \{(s_t, a_t, s_{t+1})\}$ by executing random actions in the environment.

The task-conditioned dynamics model takes the form $f_\theta(s_t, a_t, h_i)$ where θ represents the shared model parameters and $h_i \in \mathbb{R}^d$ is a learnable embedding vector for task $\mathcal{T}_i$. We initialize θ randomly and create N embedding vectors $\{h_i\}_{i=1}^N$, one for each training task. The meta-training objective jointly optimizes both components to minimize prediction error across all tasks:

$$\min_{\theta, \{h_i\}} \sum_{i=1}^N \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}_i} [\|s_{t+1} - f_\theta(s_t, a_t, h_i)\|^2] \quad (4.6)$$

We employ first-order meta-learning following the REPTILE algorithm (Nichol et al. 2018). The optimization alternates between task-specific updates and meta-parameter updates. For each training iteration, we sample a task $\mathcal{T}_i$ and perform K gradient descent steps on both the model parameters and the corresponding embedding:

$$\theta^{(k+1)} = \theta^{(k)} - \beta \nabla_{\theta} \mathcal{L}_i(\theta^{(k)}, h_i^{(k)}) \quad (4.7)$$

$$h_i^{(k+1)} = h_i^{(k)} - \beta \nabla_{h_i} \mathcal{L}_i(\theta^{(k)}, h_i^{(k)}) \quad (4.8)$$

where β is the inner-loop learning rate and $\mathcal{L}_i$ denotes the prediction error on task i . After these task-specific updates produce $\theta^{(K)}$ and $h_i^{(K)}$, we update the

meta-parameters toward their task-specific counterparts:

$$\theta \leftarrow \theta + \alpha(\theta^{(K)} - \theta) \quad (4.9)$$

$$h_i \leftarrow h_i + \alpha(h_i^{(K)} - h_i) \quad (4.10)$$

where α is the meta-learning rate.

This training procedure discovers model parameters θ that generalize across the task distribution when paired with appropriate task embeddings. Simultaneously, the learned embeddings capture distinctive characteristics of each training task in a compact representation. The joint optimization ensures that similar tasks develop similar embeddings, creating structure in the embedding space that facilitates rapid adaptation at deployment.

Among the N training tasks, we designate one as the reference task $\mathcal{T}_{\text{ref}}$ with corresponding embedding h_{ref} . After meta-training the dynamics model, we train a reference policy π_{ref} on this task using standard RL. This policy serves as the behavioral template during deployment. The choice of reference task depends on the application: it should represent conditions where appropriate operational behavior can be verified and validated before deployment to novel environments.

The meta-training phase produces three key outputs used during deployment: the meta-learned model parameters θ that enable accurate dynamics prediction across the task distribution, the reference task embedding h_{ref} that encodes validated environmental conditions, and the reference policy π_{ref} that defines behaviorally appropriate responses under those conditions. Algorithm 1 summarizes the complete meta-training procedure.

4.4.2 Reference Task Embeddings and Behavioral Constraints

The core mechanism enabling MAC to preserve appropriate behaviors lies in leveraging dual task embeddings during planning. While conventional model-based control uses learned dynamics to predict state evolution under candidate actions, MAC augments this with a behavioral reference that grounds adaptation in validated operational patterns.

The meta-trained dynamics model $f_{\theta}(s, a, h)$ conditions its predictions on task-specific embeddings $h \in \mathbb{R}^d$, enabling different dynamics to be captured for different task contexts. MAC exploits this by maintaining two embeddings simultaneously during deployment. The first, h_{new} , undergoes continuous refinement based on recent observations to track the evolving dynamics of the current environment. The second, h_{ref} , remains fixed at the value learned during meta-training for a designated reference task, preserving the dynamics under which appropriate operational behavior was validated.

Algorithm 1 Meta-Training with Task Embeddings

Require: Task distribution $p(\mathcal{T})$
Require: Meta-learning rate α , inner learning rate β
Require: Number of training tasks N , gradient steps K
Require: Embedding dimension d

- 1: Sample N training tasks: $\{\mathcal{T}_i\}_{i=1}^N \sim p(\mathcal{T})$
- 2: Collect data from each task: $\mathcal{D}_i = \{(s_t, a_t, s_{t+1})\}$ for $i = 1, \dots, N$
- 3: Initialize model parameters θ randomly
- 4: Initialize task embeddings $\{h_i \in \mathbb{R}^d\}_{i=1}^N$ randomly
- 5: **for** training iteration = 1, 2, ... **do**
- 6: Sample task $\mathcal{T}_i$ from training set
- 7: Set $\theta^{(0)} \leftarrow \theta, h_i^{(0)} \leftarrow h_i$
- 8: **for** $k = 0, 1, \dots, K - 1$ **do**
- 9: Compute loss: $\mathcal{L}_i = \mathbb{E}_{(s,a,s') \sim \mathcal{D}_i} [\|s' - f_{\theta^{(k)}}(s, a, h_i^{(k)})\|^2]$
- 10: Task-specific update θ and h_i (Equations 4.7, 4.8)
- 11: **end for**
- 12: Meta-update θ and h_i (Equations 4.9, 4.10)
- 13: **end for**
- 14: Select reference task $\mathcal{T}_{\text{ref}}$ with embedding h_{ref}
- 15: Train reference policy π_{ref} on $\mathcal{T}_{\text{ref}}$ using standard RL
- 16: **return** Meta-learned parameters θ , embeddings $\{h_i\}$, reference embedding h_{ref} , reference policy π_{ref}

By conditioning the same model on these distinct embeddings, we obtain comparative predictions at each planning step. For a given state s_t , we generate two one-step-ahead predictions using different actions and embeddings:

$$\hat{s}_{t+1}^{\text{test}} = f_{\theta}(s_t, a_t, h_{\text{new}}) \quad (4.11)$$

$$s_{t+1}^{\text{ref}} = f_{\theta}(s_t, \pi_{\text{ref}}(s_t), h_{\text{ref}}) \quad (4.12)$$

where a_t denotes the candidate action and $\pi_{\text{ref}} : \mathcal{S} \rightarrow \mathcal{A}$ is the reference policy trained through standard RL. The first prediction reveals where the candidate action would lead under current task dynamics, while the second shows where the reference policy's action would lead under reference task dynamics. Crucially, both predictions start from the same state s_t , enabling direct comparison of the resulting state transitions.

This comparison forms the basis for the behavioral constraint. When the candidate action produces a predicted state that aligns closely with the reference prediction, it indicates that the action generates movement patterns consistent with validated reference behavior, even as the underlying environmental dynamics differ. We quantify this alignment through cosine similarity:

$$\text{sim}(s, s') = \frac{\langle s, s' \rangle}{\|s\| \|s'\|} \quad (4.13)$$

The metric ranges from -1 to 1 , providing scale-independent comparison across diverse state configurations. This measure captures the directional alignment of state transitions, rather than requiring exact state matching, reflecting the insight that appropriate behaviors manifest as consistent movement patterns rather than identical absolute states across varying environmental conditions.

4.4.3 Online Model Adaptation

When MAC encounters a novel task $\mathcal{T}_{\text{new}} \sim p(\mathcal{T})$ during deployment, rapid adaptation to the new environmental dynamics becomes essential for accurate prediction and effective control. The meta-training phase produces initial parameters θ and embedding set $\{h_i\}_{i=1}^N$ that capture common structure across the task distribution. Online adaptation refines these parameters using limited interaction data collected during deployment.

We maintain an observation buffer $\mathcal{D}_{\text{adapt}} = \{(s_t, a_t, s_{t+1})\}$ containing the $\mathcal{O}$ most recent state transitions, implementing a sliding window that tracks current environmental conditions while discarding outdated observations. Initial deployment begins by identifying the most likely task embedding from the meta-training set through likelihood evaluation:

$$h_{\text{likely}} = \arg \max_{h \in \{h_i\}_{i=1}^N} \mathbb{E}_{(s,a,s') \sim \mathcal{D}_{\text{adapt}}} [\log p_{\theta}(s' | s, a, h)] \quad (4.14)$$

This initialization selects the embedding corresponding to the training task whose dynamics most closely resemble the current deployment environment, providing an informed starting point for adaptation.

Starting from the meta-learned parameters θ and the selected embedding h_{likely} , we refine both components through gradient descent on prediction error. Setting $h_{\text{new}} \leftarrow h_{\text{likely}}$, we iteratively update:

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \mathcal{L}_{\text{adapt}}(\theta, h_{\text{new}}) \quad (4.15)$$

$$h_{\text{new}} \leftarrow h_{\text{new}} - \beta \nabla_{h_{\text{new}}} \mathcal{L}_{\text{adapt}}(\theta, h_{\text{new}}) \quad (4.16)$$

where the adaptation loss measures prediction accuracy on recent observations:

$$\mathcal{L}_{\text{adapt}}(\theta, h_{\text{new}}) = \mathbb{E}_{(s,a,s') \sim \mathcal{D}_{\text{adapt}}} [\|s' - f_{\theta}(s, a, h_{\text{new}})\|^2] \quad (4.17)$$

These updates occur periodically after accumulating sufficient new observations, typically every k control steps. The joint optimization balances exploiting meta-learned structure in θ with capturing task-specific characteristics in h_{new} , providing accurate forward predictions for planning under current environmental conditions.

Algorithm 2 details the complete adaptation procedure, integrating model updates with action execution in a continuous loop throughout task deployment.

Algorithm 2 Online Adaptation with MAC

Require: Meta-learned parameters θ , embeddings $\{h_i\}_{i=1}^N$
Require: Reference embedding h_{ref} , reference policy π_{ref}
Require: Adaptation learning rate β , gradient steps G , buffer size $\mathcal{O}$

- 1: Initialize observation buffer: $\mathcal{D}_{\text{adapt}} \leftarrow \emptyset$
- 2: Collect initial observations through random exploration
- 3: Identify most likely embedding: $h_{\text{likely}} \leftarrow \arg \max_h \mathbb{E}_{\mathcal{D}_{\text{adapt}}} [\log p_\theta(s'|s, a, h)]$
- 4: Initialize adapted parameters: $\theta_{\text{adapt}} \leftarrow \theta, h_{\text{new}} \leftarrow h_{\text{likely}}$
- 5: **while** task not completed **do**
- 6: **for** $g = 1$ to G **do**
- 7: Compute loss: $\mathcal{L} \leftarrow \mathbb{E}_{\mathcal{D}_{\text{adapt}}} [\|s' - f_{\theta_{\text{adapt}}}(s, a, h_{\text{new}})\|^2]$
- 8: Update: $\theta_{\text{adapt}} \leftarrow \theta_{\text{adapt}} - \beta \nabla_\theta \mathcal{L}$
- 9: Update: $h_{\text{new}} \leftarrow h_{\text{new}} - \beta \nabla_h \mathcal{L}$
- 10: **end for**
- 11: Select action: $a_t \leftarrow \text{MAC-Control}(s_t, \theta_{\text{adapt}}, h_{\text{new}}, h_{\text{ref}}, \pi_{\text{ref}})$
- 12: Execute action, observe: $s_{t+1} \sim p(\cdot | s_t, a_t)$
- 13: Update buffer: $\mathcal{D}_{\text{adapt}} \leftarrow \mathcal{D}_{\text{adapt}} \cup \{(s_t, a_t, s_{t+1})\}$
- 14: **if** $|\mathcal{D}_{\text{adapt}}| > \mathcal{O}$ **then**
- 15: Remove oldest transition from $\mathcal{D}_{\text{adapt}}$
- 16: **end if**
- 17: $s_t \leftarrow s_{t+1}$
- 18: **end while**

4.4.4 Constrained Action Selection via Model Predictive Control

At each control timestep, MAC must select an action that advances task objectives while maintaining behavioral consistency with the reference policy. This dual objective manifests as a constrained optimization problem solved through model predictive control.

Given current state s_t , we seek an action sequence $\mathbf{a} = (a_0, \dots, a_{H-1})$ over planning horizon H that maximizes predicted cumulative reward subject to behavioral similarity constraints:

$$\begin{aligned} \max_{\mathbf{a} \in \mathcal{A}^H} J(\mathbf{a}) &= \sum_{\tau=0}^{H-1} \gamma^\tau r(s_{t+\tau}, a_\tau) \\ \text{subject to } \quad &\text{sim}(\hat{s}_{t+\tau+1}, \tilde{s}_{t+\tau+1}) \geq \delta_{\text{ref}}, \quad \tau = 0, \dots, H-1 \end{aligned} \quad (4.18)$$

where $\gamma \in [0, 1)$ denotes the discount factor, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the task reward function, and $\delta_{\text{ref}} \in [-1, 1]$ specifies the minimum acceptable similarity threshold. The predicted states $\hat{s}_{t+\tau+1}$ and reference states $\tilde{s}_{t+\tau+1}$ evolve through recursive model application as described in Equations (4.11)–(4.12).

The constraint ensures that at each step along the predicted trajectory, the state resulting from candidate actions maintains sufficient similarity to the state the

reference policy would achieve. This step-wise enforcement throughout the planning horizon prevents action sequences that might satisfy task objectives while exhibiting operationally inappropriate intermediate behaviors.

We solve this constrained optimization using the Cross-Entropy Method, a sampling-based approach particularly effective for high-dimensional continuous action spaces with non-differentiable constraints. CEM maintains a distribution $\mathcal{N}(\mu_k, \Sigma_k)$ over action sequences and iteratively refines it toward constraint-satisfying, high-reward solutions.

At each CEM iteration k , the procedure:

1. Samples N candidate action sequences from current distribution: $\{\mathbf{a}^{(j)}\}_{j=1}^N \sim \mathcal{N}(\mu_k, \Sigma_k)$
2. Evaluates each candidate through forward simulation, computing predicted reward $J(\mathbf{a}^{(j)})$ and minimum similarity $s^{(j)} = \min_{\tau} \text{sim}(\hat{s}_{t+\tau+1}^{(j)}, \tilde{s}_{t+\tau+1})$
3. Selects elite subset $\mathcal{E}_k = \{\mathbf{a}^{(j)} \mid s^{(j)} \geq \delta_{\text{ref}}\}$ satisfying constraints, retaining the $\lfloor \epsilon N \rfloor$ highest-reward members
4. Updates distribution by fitting to elite samples: $\mu_{k+1}, \Sigma_{k+1} \leftarrow \text{MLE}(\mathcal{E}_k)$

After a fixed number of iterations, the method returns the action sequence with highest reward among the final elite set. Following the receding horizon principle, only the first action from this sequence executes, and the optimization repeats at the next timestep with updated state information.

Algorithm 3 provides the complete procedure for constrained action selection, detailing the forward simulation and elite selection process.

Algorithm 3 Constrained Action Selection (MAC-Control)

Require: Current state s_t , adapted parameters $\theta_{\text{adapt}}, h_{\text{new}}$
Require: Reference embedding h_{ref} , reference policy π_{ref}
Require: Planning horizon H , similarity threshold δ_{ref}
Require: CEM parameters: population size N , elite fraction ϵ , iterations K

- 1: Initialize distribution: $\mu_0 \leftarrow \mathbf{0}, \Sigma_0 \leftarrow \sigma^2 I$
- 2: **for** $k = 1$ to K **do**
- 3: Sample candidates: $\{\mathbf{a}^{(j)}\}_{j=1}^N \sim \mathcal{N}(\mu_{k-1}, \Sigma_{k-1})$
- 4: **for** $j = 1$ to N **do**
- 5: Initialize: $s_0 \leftarrow s_t, J^{(j)} \leftarrow 0, s_{\min}^{(j)} \leftarrow 1$
- 6: **for** $\tau = 0$ to $H - 1$ **do**
- 7: Predict: $\hat{s}_{\tau+1} \leftarrow f_{\theta_{\text{adapt}}}(s_\tau, a_\tau^{(j)}, h_{\text{new}})$
- 8: Reference: $\tilde{s}_{\tau+1} \leftarrow f_{\theta_{\text{adapt}}}(s_\tau, \pi_{\text{ref}}(s_\tau), h_{\text{ref}})$
- 9: Similarity: $s_\tau^{(j)} \leftarrow \langle \hat{s}_{\tau+1}, \tilde{s}_{\tau+1} \rangle / (\|\hat{s}_{\tau+1}\| \|\tilde{s}_{\tau+1}\|)$
- 10: Reward: $J^{(j)} \leftarrow J^{(j)} + \gamma^\tau r(s_\tau, a_\tau^{(j)})$
- 11: Track minimum: $s_{\min}^{(j)} \leftarrow \min(s_{\min}^{(j)}, s_\tau^{(j)})$
- 12: Advance: $s_{\tau+1} \leftarrow \hat{s}_{\tau+1}$
- 13: **end for**
- 14: **end for**
- 15: Filter feasible: $\mathcal{F} \leftarrow \{j \mid s_{\min}^{(j)} \geq \delta_{\text{ref}}\}$
- 16: Select elites: $\mathcal{E} \leftarrow \text{top } \lfloor \epsilon N \rfloor \text{ indices in } \mathcal{F} \text{ by } J^{(j)}$
- 17: Fit distribution: $\mu_k, \Sigma_k \leftarrow \text{MLE}(\{\mathbf{a}^{(j)} \mid j \in \mathcal{E}\})$
- 18: **end for**
- 19: Select best: $j^* \leftarrow \arg \max_{j \in \mathcal{E}} J^{(j)}$
- 20: **return** First action: $a_t \leftarrow a_0^{(j^*)}$

4.5 Experimental Validation

We evaluate MAC on meta-reinforcement learning tasks requiring rapid adaptation to novel environmental conditions. The experimental validation addresses three key questions:

1. Does MAC maintain behaviorally appropriate responses when adapting to conditions outside the meta-training distribution?
2. Does the approach generalize across fundamentally different types of environmental variation?
3. How does task performance compare to model-based meta-learning baselines that lack behavioral constraints?

4.5.1 Experimental Setup

All experiments are conducted using the MuJoCo physics simulator (Todorov et al. 2012), which provides high-fidelity continuous control environments suitable for evaluating rapid adaptation under changing dynamics. We selected four locomotion scenarios representing qualitatively different sources of environmental variation: gravitational changes, morphology changes, joint blockages, and changes in limb flexibility.

Task Environments

We evaluate MAC across four distinct continuous control environments where reward maximization alone produces operationally inappropriate behaviors.

Ant-Gravity. An 8-degree-of-freedom quadruped robot must maintain forward locomotion across varying gravitational settings. The robot receives rewards proportional to forward velocity. Under low gravity, unconstrained policies discover that jumping or rolling achieves higher velocities than walking, maximizing reward while violating operational requirements for stable locomotion. The reference task uses standard gravity (9.8 m/s^2), providing validated walking behavior as the preferred operational pattern.

Ant-Disabled. The same quadruped robot encounters morphological changes through resized legs. Different leg-scaling patterns correspond to distinct tasks within the distribution. The robot must maintain forward progress despite changes to its leg geometry.

HalfCheetah-Disabled. A planar 6-degree-of-freedom robot experiences actuator failures through joint blockages that alter kinematic constraints. The robot must adapt while maintaining stable forward locomotion. The reduced dimensionality compared to the quadruped provides complementary evidence for the approach’s flexibility across different morphologies.

HalfCheetah-Pier. The same planar robot encounters changes in limb flexibility while running on moving ground. The robot must adapt to the changed dynamics while maintaining consistent gait patterns. This tests whether reference behaviors transfer effectively across dynamics rather than morphological changes.

Table 4.1 summarizes the complete task specifications, including the precise meta-training and meta-testing distributions for each environment. Notably, several meta-testing conditions involve extrapolation beyond the training range (e.g., gravity values of 3 and 25 m/s^2 versus training range of $6\text{--}17 \text{ m/s}^2$), while others test compositional generalization to entirely novel structural variations (e.g., leg 4 resized after training only on legs 1–3).

All robots receive proprioceptive observations including joint positions, velocities, and orientation information. Control occurs through continuous torque

Env.	Variation	Train tasks	Test tasks	Ref. task
Ant-Gravity	Gravity (m/s ²)	17 values in range [−17, −6]	−3, −4, −5, −10, −12, −18, −25, −30	−9.81
Ant-Disabled	Leg scaling	No resize, legs 1–3 resized	Leg 4 resized	No resize
HalfCheetah-Disabled	Joint disablement	Fixate joints 1–4	Fixate joints 5–6	No fixation
HalfCheetah-Pier	Joint damping	2 vectors*	2 vectors*	No damping

Table 4.1: Task environment specifications with meta training and meta testing configurations. Train tasks list the parameter ranges seen during meta training, while test tasks include both in-distribution configurations and **bold entries that denote extrapolation or novel settings**. The reference task specifies the nominal configuration used as a baseline in each environment.

* Refers to 6-dimensional damping coefficient vectors per configuration.

commands at 50Hz. Meta-testing episode length is fixed at 500 timesteps for all environments.

Baseline Methods

We compare MAC against two model-based meta-reinforcement learning baselines that employ identical dynamics modeling frameworks but differ in action selection mechanisms. Both baselines use MPC for planning but lack behavioral constraints.

RMPC (REPTILE + MPC). This baseline combines REPTILE meta-learning (Nichol et al. 2018) with standard MPC. The approach meta-trains a dynamics model by adapting full model parameters θ rather than using task embeddings. During deployment, MPC optimizes actions purely for reward maximization without behavioral constraints.

FMPC (FAMLE + MPC). This baseline employs the FAMLE framework (Kaushik et al. 2020) with task embeddings for efficient adaptation while using standard reward-maximizing MPC. The approach uses identical dynamics models and embedding adaptation procedures as MAC, removing only the reference embedding constraint mechanism.

All methods use identical network architectures and training hyperparameters where applicable. Complete settings, including MAC-specific parameters such as

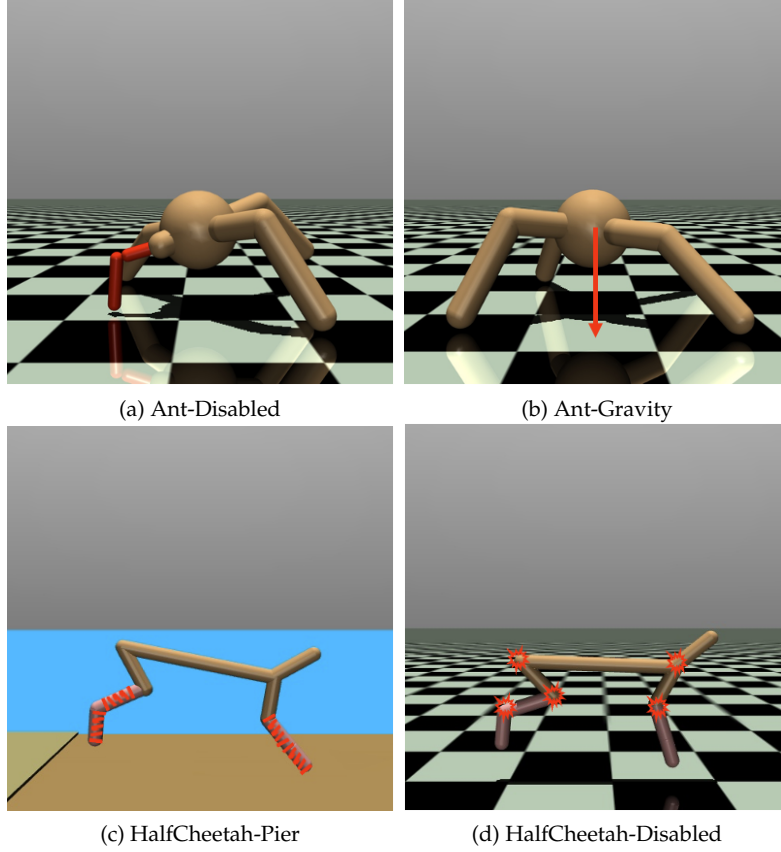


Figure 4.2: Evaluation environments for testing MAC’s behavioral constraint mechanism across different sources of environmental variation. Ant-Disabled introduces morphology changes through selectively disabled legs, while Ant-Gravity varies the gravitational acceleration. HalfCheetah-Pier adds a localized obstacle that must be crossed safely, and HalfCheetah-Disabled modifies joint actuation. Together, these environments probe MAC’s ability to handle morphology shifts, dynamics changes, and structured terrain hazards. Images adapted from Daaboul et al. (2023). © 2023 IEEE.

population size (8,192), elite count (512), and similarity threshold δ_{ref} (0.85), are provided in Appendix B.1.

For MAC specifically, the reference policy is obtained by training a Soft Actor-Critic agent (Haarnoja et al. 2018) on the reference task (standard gravity for Ant-Gravity, fully functional robot for disability tasks) until convergence.

Evaluation Protocol

Meta-Test Tasks. After meta-training converges, we evaluate on novel test tasks that assess generalization capabilities. Table 4.1 details the specific meta-testing dis-

Environment	Tasks	RMPC	FMPC	MAC
Ant-Gravity	8	23289	23957	48465
Ant-Disabled	1	2234	1856	4863
HalfCheetah-Disabled	2	4034	9439	35927
HalfCheetah-Pier	2	2956	5260	20410

Table 4.2: Meta-testing returns for MAC and baseline methods across evaluation environments. Higher values indicate better task performance. Values adapted from Daaboul et al. (2023). © 2023 IEEE.

tributions for each environment, including both interpolation within the training range and extrapolation to more extreme conditions.

Performance Metrics. We measure task performance through meta-testing return. A new task is sampled after every 500 steps, and each task is evaluated five times with different environment seeds. Table 4.2 reports returns and total evaluation steps for each environment.

Behavioral Assessment. Behavioral appropriateness is assessed qualitatively through systematic visual inspection of deployed policies. For Ant-Gravity specifically, we distinguish between appropriate walking locomotion (maintaining ground contact with coordinated leg movements) and inappropriate jumping or rolling behaviors that maximize reward while violating stability requirements.

4.5.2 Main Results

We evaluate MAC across four meta-testing environments, examining both quantitative task performance and qualitative behavioral appropriateness. The results demonstrate MAC’s ability to maintain competitive returns while preserving operationally appropriate behaviors across gravitational variations, morphological changes, and terrain modifications.

Task Performance

Table 4.2 presents meta-testing returns across all environments, evaluating performance on novel tasks not encountered during meta-training. MAC achieves higher returns than the baselines while maintaining appropriate behaviors.

MAC achieves the highest returns across all four environments, demonstrating that behavioral constraints do not negatively impact task performance when the constraint aligns with effective task completion strategies. This consistent performance across gravitational variations, morphological changes, and terrain

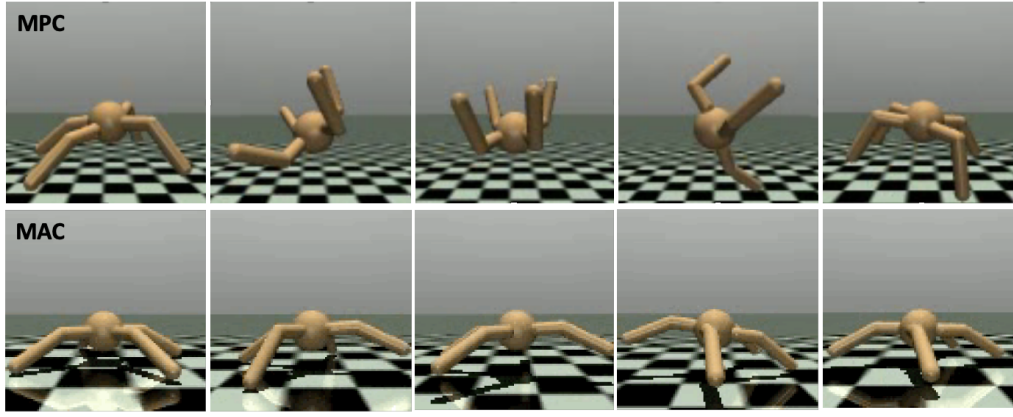


Figure 4.3: Behavioral comparison during meta testing on Ant-Gravity across different control strategies. Baseline methods that focus primarily on return often exploit unsafe but high reward behaviors, for example by rolling or tumbling in ways that violate task preferences. In contrast, MAC yields gaits that remain closer to the intended walking behavior, maintaining stability while still achieving high performance. The comparison illustrates how encoding behavioral preferences in the meta learning objective helps MAC prioritize operational appropriateness over pure reward maximization. Reproduced from Daaboul et al. (2023). © 2023 IEEE.

modifications indicates that the reference embedding mechanism generalizes effectively beyond specific variation types without requiring environment-specific modifications. The consistent superiority of embedding-based methods (MAC and FMPC) over REPTILE-based adaptation (RMPC) across most environments further validates the effectiveness of learned task embeddings for rapid adaptation under changing dynamics.

Behavioral Appropriateness

Beyond numerical performance, the qualitative differences in adapted behaviors reveal the practical value of MAC. Figure 4.3 illustrates the contrast between MAC and unconstrained MPC in the Ant-Gravity environment under reduced gravitational forces. RMPC discovers that jumping and rolling motions achieve higher forward velocities than walking gaits, maximizing the reward function while violating fundamental operational requirements for stable locomotion.

MAC successfully prevents these inappropriate adaptations through its behavioral similarity constraint mechanism. By requiring that predicted trajectories maintain cosine similarity above threshold δ_{ref} with trajectories generated under the reference task embedding, MAC preserves walking locomotion even under novel gravitational settings. The adapted policy maintains ground contact and

coordinated leg movements consistent with the reference behavior, demonstrating effective transfer of validated movement patterns.

In the reported experiments, this behavioral constraint does not reduce task performance: MAC achieves the highest return in Ant-Gravity while maintaining walking locomotion (Table 4.2).

The MAC framework demonstrates that rapid adaptation and behavioral appropriateness can be achieved simultaneously through similarity-constrained model predictive control. By leveraging task embeddings to encode both current dynamics and preferred behaviors, MAC enables robots to maintain operational safety and consistency while responding to novel environmental conditions. However, several limitations and open challenges merit discussion.

4.6 Discussion and Limitations

4.6.1 Behavioral Appropriateness and Performance

The experimental results demonstrate that task rewards and operationally appropriate behaviors can be improved together. A key advantage of MAC’s reference embedding approach lies in its ability to encode preferred behaviors through demonstration rather than requiring task-specific reward engineering. Where traditional methods would need carefully designed penalty terms to discourage jumping or rolling in each environment, MAC learns these preferences implicitly from the reference task, generalizing behavioral constraints across novel conditions without explicit specification.

This approach proves effective across all four evaluation scenarios. MAC achieves the highest returns while preserving appropriate behaviors, indicating that behavioral constraints can align naturally with effective task completion strategies.

4.6.2 Generalization Across Environmental Variations

MAC’s consistent performance across gravitational changes, morphological variations, and terrain modifications indicates that the reference embedding mechanism successfully transfers behavioral constraints beyond specific variation types. The approach generalizes without environment-specific modifications, demonstrating that learned task embeddings capture behavioral patterns at a level of abstraction that transcends particular sources of dynamics variation. Notably, MAC maintains appropriate behaviors even under extrapolation conditions substantially outside the meta-training distribution, such as gravitational accelerations of -3 m/s^2 and -30 m/s^2 compared to the training range of $[-17, -6] \text{ m/s}^2$. This extrapolation capability suggests that the embedding space encodes meaningful structure about dynamics variation rather than merely memorizing training configurations.

The success across different robot morphologies (quadrupedal Ant and planar HalfCheetah) further validates the approach’s flexibility.

4.6.3 Limitations

MAC’s effectiveness depends critically on selecting an appropriate reference task that exhibits genuinely preferred behaviors. This requirement becomes problematic when different deployment scenarios demand fundamentally different behavioral patterns. For instance, if aggressive maneuvers prove appropriate in some conditions but conservative movements in others, no single reference task adequately captures context-dependent preferences. The framework currently assumes one behavioral pattern generalizes across the entire task distribution, limiting applicability when operational requirements vary substantially between tasks.

The cosine similarity metric, while computationally efficient, treats all state dimensions as equally important for behavioral appropriateness. In practice, certain aspects of behavior matter far more than others. Maintaining torso orientation may be critical for operational safety, while exact foot placement patterns prove less consequential. The current formulation cannot distinguish between these priorities, potentially constraining irrelevant dimensions while inadequately preserving essential behavioral characteristics. Future work should explore learned similarity metrics that automatically identify and emphasize operationally critical state dimensions.

Additional limitations include scalability to complex task distributions requiring larger embedding spaces and model capacity, and computational constraints for high-frequency control applications where planning overhead becomes prohibitive.

5 Constrained Model-Agnostic Meta Learning

5.1 Overview

Meta-reinforcement learning enables rapid adaptation to new conditions by learning effective initializations. While Chapter 4 demonstrated this principle for adapting dynamics models to environmental changes, the present chapter addresses adaptation of policy parameters to new tasks through meta-learned initializations that facilitate quick fine-tuning with minimal experience. Consider a rescue robot meta-trained to locate and reach people in dangerous areas across various building layouts and hazard configurations. When deployed in a new disaster scenario with unfamiliar obstacle placements and victim locations, standard meta-learning enables the robot to adapt its search and navigation strategy quickly. However, this rapid adaptation may produce trajectories that collide with unstable debris, joint configurations that exceed safe torque limits, or behaviors that approach workspace boundaries. The meta-learned initialization optimizes for fast task adaptation. However, it provides no mechanism to ensure that the resulting behaviors remain within operational safety bounds throughout both the adaptation process and deployment.

This chapter introduces CMAML, a framework that integrates safety constraints directly into gradient-based meta-learning. Rather than treating safety as an external consideration imposed after learning, CMAML embeds constraint satisfaction into the meta-objective itself, learning initializations that simultaneously enable efficient adaptation and guarantee safety. Where Chapter 3 achieved sample-efficient safe learning within individual tasks through model-based methods, this work addresses a fundamentally different challenge: maintaining safety across entire task distributions during meta-training while ensuring safe behavior during rapid adaptation to novel scenarios.

The approach combines three mechanisms. Task-specific safety during adaptation relies on constrained policy optimization algorithms, such as TRPOLag, in the inner loop, ensuring that each policy update respects local constraints. Meta-policy safety across the distribution emerges from an additional global constraint in the outer loop, aggregating requirements from all tasks to prevent drift toward parameter regions that sacrifice safety for adaptation speed. Computational efficiency stems from first-order meta-gradient approximations combined with a learned

meta-cost critic. Trained on experience collected by the meta-policy across all encountered tasks, this critic guides outer-loop optimization toward inherently safer parameter regions without requiring expensive second-order derivatives.

This chapter proceeds as follows. Section 5.2 positions the work relative to existing meta-learning and safe reinforcement learning research. Section 5.3 establishes the problem formulation and necessary background. Section 5.4 presents the algorithm in detail, covering constrained inner-loop optimization, the safety-augmented outer-loop objective with meta cost critic, and first-order gradient approximations. Section 5.5 validates the approach through experiments demonstrating that CMAML produces meta-policies that adapt quickly while maintaining safety during both meta-training and fine-tuning. Section 5.6 concludes with limitations and future directions.

5.2 Related Work

Previous chapters established the necessary foundations for this work. Chapter 3 reviewed safe reinforcement learning through CMDPs, CPO, and Lagrangian optimization, demonstrating how ensemble-based uncertainty quantification enables sample-efficient exploration under constraints. Chapters 2 and 4 established the meta-learning foundations, from MAML, and first-order variants to context-based techniques such as FAMLE, showing how learned initializations facilitate rapid adaptation to new conditions.

These research directions have mainly evolved in isolation. Safe reinforcement learning operates within fixed task contexts, assuming constraints remain constant throughout training. Meta-learning pursues rapid adaptation across task distributions but provides no mechanism to enforce safety during this process. The result is a fundamental gap: existing methods cannot simultaneously guarantee fast adaptation and constraint satisfaction across diverse scenarios with varying objectives and safety requirements.

CMAML addresses this through constrained bi-level optimization that embeds safety at both the task-specific and distribution levels. The framework ensures local constraint satisfaction during inner-loop adaptation while maintaining global safety properties through the meta-policy, achieving rapid fine-tuning without sacrificing safety guarantees.

5.2.1 Safe Meta-Reinforcement Learning

Safe meta-reinforcement learning is still relatively underexplored compared to unconstrained meta-RL. Early work largely treated safety as a post-hoc modification, for example, by training a fast-adapting meta-policy in the standard way and then wrapping it with a separate safety filter or shield (García and Fernández

2015). Such approaches can reduce obvious failures but provide weak guarantees when the adapted policy encounters truly novel safety-critical situations, since the meta-initialization itself is not safety-aware.

More recent methods integrate safety directly into the meta-learning process, but each does so under restrictive assumptions. Curriculum-based safe RL learns task-specific safe sets by intervening during training (Turchetta et al. 2020), which yields localized guarantees but struggles when the deployment task has constraint structures that differ from those seen in the curriculum. Online constrained meta-learning frameworks provide convergence guarantees for both performance and constraint violation, yet typically assume shared cost functions or known constraint thresholds across tasks (Xu and Zhu 2023). Offline safe meta-RL, such as cost-aware meta-learning with safety-informed context encoders (Guan et al. 2024), can improve robustness within the support of the offline data but requires broad and diverse task coverage to generalize to unseen constraints. Finally, differentiable convex-programming-based meta-RL (Cho and Sun 2024) achieves end-to-end trainable safety guarantees, but is limited to problems that admit convex cost functions and known constraint parametrizations. As a result, achieving scalable, safety-aware meta-initializations that reliably generalize to novel tasks and constraint structures remains an open challenge.

5.3 Background and Problem Formulation

We build upon the constrained reinforcement learning framework established in Chapter 3 and the meta-learning foundations from Chapter 4. This section introduces the constrained meta-learning problem formulation and establishes the theoretical foundations for CMAML.

5.3.1 Gradient-Based Meta-Learning

The meta-learning framework from Section 2.3.2 establishes how to learn initializations that enable rapid task adaptation. CMAML implements this through Model-Agnostic Meta-Learning (MAML) (Finn et al. 2017), which optimizes meta-parameters θ via bi-level optimization: an inner loop that adapts the generalist meta-policy to become a specialized task-specific policy, and an outer loop that updates θ to improve the meta-policy’s adaptability across the task distribution.

For task $\mathcal{T}_i$ with objective function J_i measuring expected return, the inner loop performs K gradient steps of adaptation starting from the meta-parameters θ . Beginning from $\theta_i^{(0)} = \theta$, each adaptation step updates:

$$\theta_i^{(k+1)} = \theta_i^{(k)} + \alpha_{\text{inner}} \nabla_{\theta_i^{(k)}} J_i(\theta_i^{(k)}) \quad (5.1)$$

for $k = 0, 1, \dots, K - 1$, where α_{inner} is the inner loop learning rate. This produces task-specific parameters $\theta_i^{(K)}$. In subsequent sections, we drop the (K) superscript and simply write θ_i to denote these final adapted parameters.

The meta-objective optimizes the meta-parameters to maximize expected post-adaptation performance across the task distribution:

$$\max_{\theta} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} [J_i(\theta_i)] \quad (5.2)$$

Computing the meta-gradient requires understanding how this meta-objective changes with respect to θ . The meta-gradient decomposes as:

$$\nabla_{\theta} J_i(\theta_i) = \nabla_{\theta_i} J_i(\theta_i) \cdot \frac{\partial \theta_i}{\partial \theta} \quad (5.3)$$

where $\nabla_{\theta_i} J_i(\theta_i)$ represents how the objective changes with respect to the adapted parameters, and $\frac{\partial \theta_i}{\partial \theta}$ captures how those adapted parameters depend on the meta-initialization.

The critical computational challenge lies in the second term. Computing $\frac{\partial \theta_i}{\partial \theta}$ requires differentiating through all K inner loop gradient steps, yielding second-order derivatives (Hessians) with respect to θ . For high-dimensional policy networks with millions of parameters, this computation becomes prohibitively expensive in both memory and time.

First-Order MAML (First-Order Model-Agnostic Meta-Learning (FOMAML)) (Finn et al. 2017) addresses this by approximating the meta-gradient using only first-order information. Rather than computing the full chain rule, FOMAML treats the adapted parameters θ_i as constants when computing the outer loop gradient:

$$\nabla_{\theta} J_i(\theta_i) \approx \nabla_{\theta_i} J_i(\theta_i) \quad (5.4)$$

This approximation ignores how θ_i depends on θ , eliminating the need for second-order derivatives. Despite discarding this Hessian information, FOMAML maintains effective meta-learning. Gradients from the specialized task policies still flow back to update the generalist meta-policy θ in directions that facilitate rapid adaptation. The computational savings prove substantial: FOMAML reduces the cost from $O(K \cdot d^2)$ to $O(d)$ per task, where d represents the number of policy parameters and K is the number of inner loop steps.

CMAML employs FOMAML in the outer loop optimization to make constrained meta-learning tractable. However, the first-order approximation discards information about how constraint satisfaction changes during adaptation. The meta safety critic introduced in CMAML compensates for this by providing explicit safety signals during meta-parameter updates, ensuring that θ leads to both rapid adaptation and reliable constraint satisfaction.

5.3.2 Constrained Policy Optimization via Lagrangian Relaxation

Chapter 3 established the constrained reinforcement learning problem where policies must maximize expected return while satisfying cost constraints. For a single constraint, the optimization problem takes the form:

$$\max_{\pi} J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \quad (5.5)$$

$$\text{subject to } J_c(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t c_t \right] \leq d \quad (5.6)$$

where $J(\pi)$ denotes the expected return, $J_c(\pi)$ denotes the expected cumulative cost, and d is the cost threshold.

Lagrangian methods address this by introducing a dual variable $\lambda \geq 0$ that transforms the constrained problem into an unconstrained optimization (Boyd and Vandenberghe 2004). The Lagrangian function $\mathcal{L}(\pi, \lambda)$ combines the objective with a penalty for constraint violation:

$$\mathcal{L}(\pi, \lambda) = J(\pi) - \lambda(J_c(\pi) - d) \quad (5.7)$$

The optimal policy satisfies the saddle point condition:

$$\pi^* = \arg \min_{\lambda \geq 0} \max_{\pi} \mathcal{L}(\pi, \lambda) \quad (5.8)$$

This leads to primal-dual optimization where policy parameters and the dual variable are updated iteratively. The policy update maximizes the Lagrangian function (5.7) with respect to the policy parameters:

$$\theta \leftarrow \theta + \alpha_{\text{primal}} \nabla_{\theta} \mathcal{L}(\pi_{\theta}, \lambda) \quad (5.9)$$

The dual variable adjusts based on constraint satisfaction:

$$\lambda \leftarrow \max(0, \lambda + \alpha_{\text{dual}}(J_c(\pi_{\theta}) - d)) \quad (5.10)$$

When costs exceed the threshold, λ increases and places greater emphasis on cost reduction in subsequent policy updates. When costs remain below the threshold, λ decreases, allowing the policy to focus more on reward maximization. This adaptive penalty mechanism guides the policy toward the constraint boundary while maintaining feasibility.

Trust Region Policy Optimization with Lagrangian Methods

Trust Region Policy Optimization with Lagrangian methods (TRPOLag) (Ray et al. 2019) combines the Lagrangian framework with TRPO’s trust region optimization. As established in Section 3.3.2, TRPO optimizes a surrogate objective $L_\pi(\pi')$ that approximates the true expected return $J(\pi')$ while constraining policy updates through KL divergence:

$$\max_{\pi'} L_\pi(\pi') \quad \text{subject to} \quad \mathbb{E}_{s \sim d^\pi} [D_{\text{KL}}(\pi'(\cdot|s) \parallel \pi(\cdot|s))] \leq \delta \quad (5.11)$$

TRPOLag extends this to constrained problems by replacing the surrogate objective with a surrogate Lagrangian. Analogous to the reward surrogate, we define a cost surrogate $L_c(\pi')$ that approximates $J_c(\pi')$:

$$L_c(\pi') = \mathbb{E}_{s \sim d^\pi, a \sim \pi} \left[\frac{\pi'(a|s)}{\pi(a|s)} A_c^\pi(s, a) \right] \quad (5.12)$$

where $A_c^\pi(s, a)$ is the cost advantage function. The surrogate Lagrangian then becomes:

$$\mathcal{L}(\pi', \lambda) = L_\pi(\pi') - \lambda(L_c(\pi') - d) \quad (5.13)$$

At each iteration, TRPOLag solves the trust region subproblem:

$$\max_{\pi'} \mathcal{L}(\pi', \lambda) \quad \text{subject to} \quad \mathbb{E}_{s \sim d^\pi} [D_{\text{KL}}(\pi'(\cdot|s) \parallel \pi(\cdot|s))] \leq \delta \quad (5.14)$$

This optimization is solved using conjugate gradient methods combined with line search, following the approach described in Section 3.3.2. Following each policy update, the dual variable λ is adjusted according to Equation (5.10).

The trust region constraint ensures stable policy updates, while the adaptive dual variable guides the policy toward feasible solutions. This combination provides both monotonic improvement guarantees and asymptotic convergence to constraint-satisfying policies, making TRPOLag well-suited for the inner loop optimization in constrained meta-learning.

5.4 Method

The CMAML framework learns policy parameters θ that enable rapid adaptation to new constrained tasks while maintaining safety throughout the learning process. Each task $\mathcal{T}_i$ in the distribution $p(\mathcal{T})$ represents a CMDP with its own reward structure, cost function, and safety threshold. The challenge lies in discovering meta-parameters that facilitate quick specialization to these diverse tasks without relying on unsafe adaptation strategies.

The framework operates through two complementary mechanisms. First, task-specific adaptation through TRPOLag ensures that each specialized policy satisfies its local safety requirements after K gradient steps. Second, a meta safety critic evaluates the meta-policy itself across the task distribution, providing safety signals that shape the meta-parameters toward initializations that exhibit inherently safe behavior before any adaptation occurs. This dual approach addresses safety at both the task level (after specialization) and the distribution level (before specialization), preventing the meta-learning process from discovering initializations that enable fast adaptation through systematic constraint violations.

The following subsections develop this framework systematically. We begin by formalizing the constrained meta-learning problem, establishing the distinction between task-level and distribution-level safety. We then detail how the inner loop adapts to individual tasks while respecting local constraints, and how the outer loop shapes meta-parameters using both adapted policy gradients and critic-based safety signals.

5.4.1 Constrained Meta-Learning Problem

The constrained meta-learning problem extends standard MAML by introducing safety requirements that must hold across two levels of the learning hierarchy. Tasks in the distribution share common structure: the same state space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $\mathcal{P}$, and discount factor γ . However, they differ in their reward functions r_i , cost functions c_i , and safety thresholds d_i . This structure captures realistic deployment scenarios where the robot’s physical capabilities remain constant but operational objectives and safety requirements change across different environments or mission profiles.

At the task level, each adaptation must produce a specialized policy that maximizes returns while maintaining operational safety for that specific task:

$$\theta_i = \arg \max_{\theta'} J_i(\pi_{\theta'}) \quad \text{subject to} \quad J_{c_i}(\pi_{\theta'}) \leq d_i \quad (5.15)$$

where $J_i(\pi_{\theta'})$ denotes the expected return and $J_{c_i}(\pi_{\theta'})$ denotes the expected cumulative cost on task $\mathcal{T}_i$. This ensures that after adaptation completes, the resulting policy π_{θ_i} operates safely within the constraints of its specific deployment scenario.

At the distribution level, the meta-parameters must optimize for expected post-adaptation performance while preventing safety violations in aggregate:

$$\max_{\theta} \quad \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} [J_i(\pi_{\theta_i})] \quad (5.16)$$

$$\text{subject to} \quad \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} [J_{c_i}(\pi_{\theta_i}) - d_i] \leq 0 \quad (5.17)$$

Equation (5.17) aggregates safety requirements across all tasks, requiring that the expected constraint violation remains non-positive when averaged over the entire

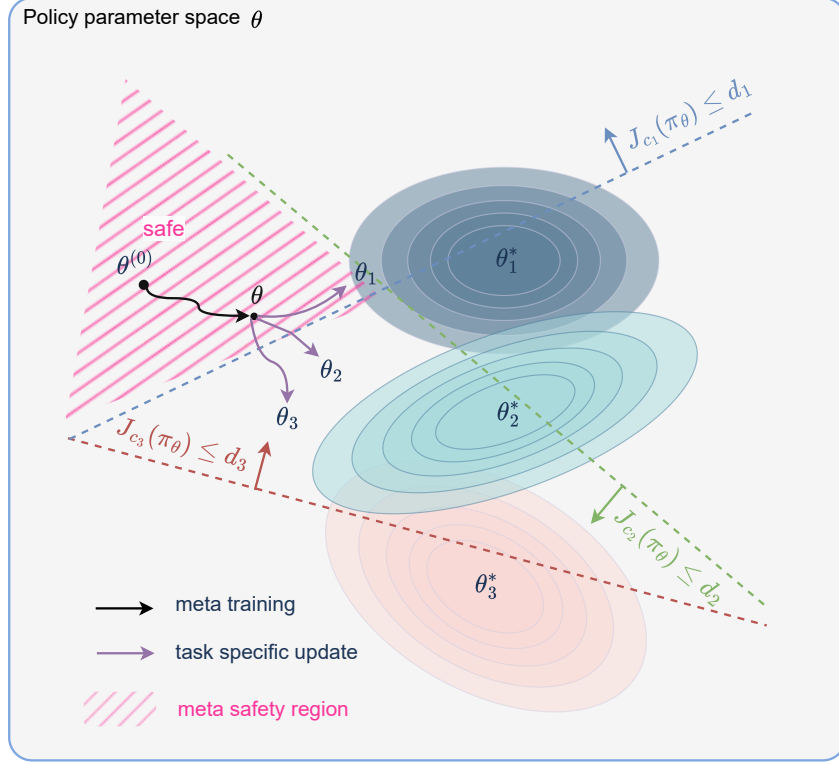


Figure 5.1: Constrained meta-learning in policy parameter space θ . Meta-training updates the parameters from an initial random initialization $\theta^{(0)}$ to a safe meta-initialization θ (black curve) that lies in the meta safety region (pink hatched area), defined as the intersection of the task-specific safe half-spaces. For each task $\mathcal{T}_i$, the inner loop starts from θ and performs constrained adaptation (violet arrows) to obtain task-specific parameters θ_i , which move toward the task optima θ_i^* inside the corresponding return landscapes. The dashed lines show the constraint boundaries $J_{c_i}(\pi_\theta) = d_i$, and the small arrows orthogonal to each line indicate the feasible side where $J_{c_i}(\pi_\theta) \leq d_i$.

distribution. This prevents a subtle failure mode: the meta-learner might discover meta-parameters that enable fast adaptation by sacrificing safety on some tasks to gain performance on others, satisfying local constraints task by task while violating safety requirements in expectation. Figure 5.1 provides a geometric illustration of this bi-level constrained meta-learning problem in policy parameter space, showing the meta safety region, task-specific optima, and constraint boundaries.

The distinction between these two levels proves crucial. A meta-policy that produces safe adapted policies does not necessarily exhibit safe behavior itself when deployed without adaptation. Conversely, a safe meta-policy might adapt into unsafe specialized policies if the adaptation process prioritizes reward maximization without maintaining safety awareness. CMAML must therefore address both

levels, ensuring that the initialization encodes safe behavior while the adaptation process preserves this safety throughout specialization.

5.4.2 Inner Loop: Task-Specific Constrained Optimization

For each sampled task $\mathcal{T}_i$, the inner loop performs constrained policy optimization to obtain task-specific parameters θ_i that satisfy local safety requirements. We employ TRPOLag (Section 5.3.2), which combines trust region optimization with Lagrangian constraint handling.

Starting from the meta-parameters θ , we perform K iterative policy updates on task $\mathcal{T}_i$. The task-specific Lagrange multiplier λ_i initializes to zero. At each iteration, we collect trajectories using the current policy, compute reward and cost advantages, and solve the trust region optimization from Equation (5.14) adapted to task i with task-specific surrogates.

Following each policy update, the Lagrange multiplier adjusts based on the observed constraint violation:

$$\lambda_i \leftarrow \max(0, \lambda_i + \alpha_{\text{dual}}(J_{c_i}(\pi_{\theta'}) - d_i)) \quad (5.18)$$

After K iterations, the adapted parameters θ_i balance task-specific returns against the cost threshold d_i , providing the task-specific gradients for the outer loop meta-update.

5.4.3 Outer Loop: Meta-Policy Optimization with Safety Critic

The outer loop optimizes meta-parameters θ according to the constrained optimization in Equations (5.16) and (5.17). Following the FOMAML approach from Section 5.3, we treat adapted parameters θ_i as constants when computing meta-gradients. For a batch of N sampled tasks, the meta-gradient is:

$$g_R = \frac{1}{N} \sum_{i=1}^N \nabla_{\theta_i} J_i(\pi_{\theta_i}) \quad (5.19)$$

The first-order approximation discards second-order information that captures how the meta-initialization influences the adaptation trajectory. During the K inner loop steps, the policy navigates through parameter space guided by both reward objectives and constraint penalties. The second-order term $\frac{\partial \theta_i^{(K)}}{\partial \theta}$ would reveal how different meta-initializations lead to different adaptation paths with varying constraint satisfaction along the way. By treating adapted parameters θ_i as constants, the FOMAML gradient reflects constraint violations only at the final adapted policy, not during the adaptation process itself. The meta-policy

may therefore learn initializations that produce safe specialists but do so through adaptation paths that systematically violate constraints during specialization.

We address this limitation by introducing a second constraint that evaluates meta-policy safety directly across the task distribution. This yields a bi-level constrained optimization problem:

$$\max_{\theta} \quad \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} [J_i(\pi_{\theta_i})] \quad (5.20)$$

$$\text{subject to} \quad \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} [J_{c_i}(\pi_{\theta_i}) - d_i] \leq 0 \quad (5.21)$$

$$\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} [J_{c_i}(\pi_{\theta}) - d_i] \leq 0 \quad (5.22)$$

Equation (5.21) ensures that adapted policies satisfy their constraints after specialization, while Equation (5.22) ensures the meta-policy itself exhibits safe behavior before any adaptation occurs. These constraints operate at different levels of the learning hierarchy: the first captures the safety of task-specific specialists, while the second captures the safety of the generalist initialization. Together, they prevent the meta-learning process from discovering initializations that enable fast adaptation at the cost of safety during the specialization process.

Applying Lagrangian relaxation with dual variables λ and η to both constraints yields the meta-objective:

$$\begin{aligned} \mathcal{L}_{\text{meta}}(\theta, \lambda, \eta) = \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} & \left[J_i(\pi_{\theta_i}) \right. \\ & - \lambda(J_{c_i}(\pi_{\theta_i}) - d_i) \\ & \left. - \eta(J_{c_i}(\pi_{\theta}) - d_i) \right] \end{aligned} \quad (5.23)$$

The dual variable λ regulates constraint satisfaction across adapted policies by adjusting the penalty for post-adaptation violations. The dual variable η regulates meta-policy safety by penalizing initializations that exhibit high expected costs when deployed directly across the task distribution. This dual regulation mechanism ensures that meta-parameters encode safe behavior from the start while maintaining safety through adaptation.

Evaluating the meta-policy constraint $\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} [J_{c_i}(\pi_{\theta})]$ through direct trajectory sampling would require collecting rollouts under π_{θ} for each sampled task at every meta-update, which proves computationally prohibitive. We instead employ a meta safety critic $V_C^{\theta}(s)$ that learns to estimate expected costs under the meta-policy from experience accumulated across all encountered tasks during meta-training.

5.4.4 Meta Safety Critic Training and Integration

The meta safety critic $V_C^{\theta}(s)$ estimates expected cumulative costs from any state when following the meta-policy:

$$V_C^{\theta}(s) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^{\infty} \gamma^t c_t \mid s_0 = s \right] \quad (5.24)$$

This critic trains via temporal difference learning on experience accumulated as the meta-policy interacts with sampled tasks during meta-training. As π_θ encounters diverse deployment scenarios, we collect transitions (s_t, a_t, c_t, s_{t+1}) from across the task distribution. The critic updates to minimize temporal difference error:

$$\mathcal{L}_C = \mathbb{E}_{s,s' \sim \mathcal{D}} [(V_C^\theta(s) - (c(s, a) + \gamma V_C^\theta(s')))^2] \quad (5.25)$$

where $\mathcal{D}$ contains transitions from all tasks encountered during meta-training.

The choice to train exclusively on meta-policy data rather than adapted policy trajectories proves essential for distribution-level safety estimation. Adapted policies exhibit task-specific behaviors that may differ substantially from the meta-policy’s behavior on the same tasks. A critic trained on adapted policy data would learn to predict costs under specialized behavior, providing inaccurate estimates for the meta-policy constraint in Equation (5.22). By training only on meta-policy rollouts, the critic learns to predict costs under the actual distribution of states and actions generated by π_θ before any adaptation occurs. This enables the critic to capture safety patterns that persist across tasks, identifying states and actions that consistently lead to high costs regardless of specific task configurations.

Integrating the critic into the meta-objective replaces the intractable expectation in Equation (5.23) with the critic’s learned estimate:

$$\begin{aligned} \mathcal{L}_{\text{meta}}(\theta, \lambda, \eta) = \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \Big[& J_i(\pi_{\theta_i}) \\ & - \lambda(J_{c_i}(\pi_{\theta_i}) - d_i) \\ & - \eta(V_C^\theta(s) - d_i) \Big] \end{aligned} \quad (5.26)$$

Computing the gradient with respect to meta-parameters yields two complementary components. The first derives from FOMAML, balancing rewards and costs across adapted policies:

$$g_{\text{FoMAML}} = \frac{1}{N} \sum_{i=1}^N \nabla_{\theta_i} [J_i(\pi_{\theta_i}) - \lambda(J_{c_i}(\pi_{\theta_i}) - d_i)] \quad (5.27)$$

The second derives from the meta safety critic, operating independently of task-specific adaptations:

$$g_{\text{critic}} = \eta \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t \log \pi_\theta(a_t | s_t) V_C^\theta(s_t) \right] \quad (5.28)$$

This critic gradient implements policy gradient weighted by predicted costs. In states where the critic estimates high cumulative costs, the gradient decreases the likelihood of actions chosen by the meta-policy. In states where the critic predicts low costs, the gradient increases action likelihoods. This mechanism directly shapes meta-policy behavior, eliminating the need for differentiation through the

K inner loop adaptation steps, thereby maintaining computational tractability under the first-order approximation while providing explicit safety signals that complement the adapted policy gradients.

The complete meta-parameter update combines both gradient sources:

$$\theta \leftarrow \theta + \alpha(g_{\text{FoMAML}} - g_{\text{critic}}) \quad (5.29)$$

where α is the meta-learning rate. The dual variables update based on observed violations: λ adjusts according to constraint satisfaction of adapted policies, while η adjusts according to meta-policy costs estimated by the critic. Through this combined optimization, CMAML learns initializations that enable rapid task-specific adaptation while maintaining safety throughout the specialization process.

5.5 Experimental Validation

This section evaluates CMAML on constrained robotic navigation tasks in the SafetyGymnasium environment. We examine mobile robot navigation across 106 tasks with varying obstacle configurations and goal locations while maintaining consistent safety requirements. The experimental validation addresses three central questions:

1. Does the meta safety critic improve constraint satisfaction during meta-training and fine-tuning compared to optimization without the additional safety mechanism?
2. How does CMAML compare to meta-learning and safe reinforcement learning baselines in adaptation speed and cost control?
3. Does the framework remain effective when different constrained policy optimization methods are employed in the inner loop?

5.5.1 Experimental Setup

We evaluate CMAML on constrained mobile robot navigation tasks requiring rapid adaptation to novel obstacle configurations while maintaining consistent safety constraints. The experiments use the SafetyGymnasium benchmark (Ji et al. 2023), which provides standardized cost functions and reward structures for evaluating safe reinforcement learning in continuous control settings.

The evaluation centers on a spherical mobile robot navigating toward goal regions while avoiding hazardous obstacles in a two-dimensional workspace. The robot controls translational velocity and rotational orientation through a continuous two-dimensional action space. Observations include position, velocity, orientation, and LiDAR-based distance measurements to nearby obstacles, providing sufficient

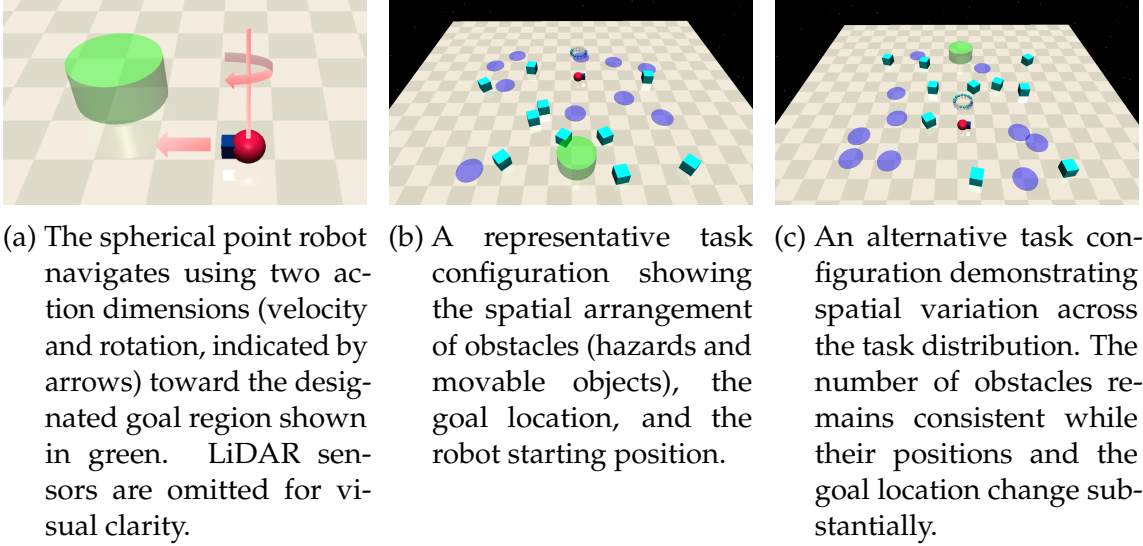


Figure 5.2: The SafetyGymnasium (Ji et al. 2023) navigation environment showing the robot action space (left) and two distinct task configurations from the 106 task distribution (center and right), illustrating the variation in obstacle placement and goal positioning.

information for collision avoidance without requiring complete knowledge of the environment.

The reward structure provides positive reinforcement proportional to the distance reduction toward the goal at each timestep. Episodes terminate when the robot reaches the goal zone or after 1000 timesteps. Cost penalties accumulate through contact with hazards or displacement of movable objects, with each collision contributing to cumulative episode cost.

The task distribution encompasses 106 distinct navigation scenarios, each with unique obstacle positions, goal locations, and robot starting states. While the obstacle count remains constant, the spatial arrangements vary substantially, requiring different navigation strategies for each configuration. Figure 5.2 illustrates representative instances demonstrating the variation defining the meta-learning challenge. This structure creates a natural setting for evaluating constrained meta-learning: a consistent cost function across tasks represents shared safety requirements, while varying spatial arrangements demand rapid adaptation of navigation strategies.

Baseline Methods

We compare CMAML against three baselines, isolating different aspects of meta-learning and safety integration.

Random Initialization. Policy parameters are initialized randomly from a standard normal distribution. This baseline establishes the lower bound on adaptation

performance, demonstrating the value of meta-learning for rapid adaptation without prior knowledge.

Pretrained Safe Policy. A policy trained with TRPOLag (Ray et al. 2019) on a single task until convergence provides initialization for novel tasks. This captures traditional transfer learning where task-specific pretraining substitutes for meta-learning across distributions.

Unconstrained MAML. Standard MAML (Finn et al. 2017) with TRPO (Schulman et al. 2015) in the inner loop learns rapid adaptation without encoding safety into meta-parameters. This isolates the effect of integrating constraints into meta-learning objectives.

CMAML. Our method extends MAML by incorporating safety constraints into both optimization loops. The inner loop uses TRPOLag for constraint-respecting adaptation, while the outer loop optimizes post-adaptation performance subject to a meta-level safety constraint regulated by trainable Lagrange multiplier η . The meta-safety critic provides learned violation estimates that guide outer-loop optimization.

All methods employ identical architectures, consisting of two hidden layers with 64 units and tanh activations. During meta-training, we sample batches of 10 tasks with five gradient steps per inner loop adaptation. The outer loop uses a learning rate of 0.001. Fine-tuning performs 100 adaptation steps per task, averaged over five random seeds. Appendix C.1 provides complete specifications.

To demonstrate algorithm agnosticity, we replace TRPOLag with CPO (Achiam et al. 2017) in both inner loop and fine-tuning, testing whether benefits generalize beyond specific implementations.

Evaluation Protocol

The evaluation separates meta-training effectiveness from fine-tuning performance, examining how well meta-learned initializations encode safety-aware adaptation and how strategies transfer to novel configurations.

Meta-Training Phase. All methods meta-train on 106 navigation tasks. Each iteration samples 10 tasks, performs five-step inner loop adaptation, and updates meta-parameters based on post-adaptation performance. We monitor mean episode return (task performance) and mean episode cost (constraint satisfaction) for 1000 iterations. For CMAML, we track the Lagrange multiplier η evolution to verify the safety critic regulation.

Fine-Tuning Phase. After meta-training, we evaluate rapid adaptation to the same 106 tasks from meta-learned initializations. Each trial begins with appropriate initialization and performs 100 gradient steps using the meta-training algorithm. We measure episode return and cost every 10 steps throughout adaptation.

Performance Metrics. Episode return measures cumulative reward during navigation. The episode cost sums penalties from collisions and object displacement. High returns indicate effective goal-reaching; costs below thresholds indicate successful constraint satisfaction. For TRPOLag experiments, the cost threshold is $d = 5$; for CPO, we use $d = 10$ to account for algorithmic differences in violation handling.

All results are averages over five independent trials with different random seeds. We report means with standard deviation bands indicating consistency. For adaptive η versus fixed $\eta = 0$ comparisons, we use paired evaluations on identical task sequences.

This structure systematically addresses our three questions: meta-training analysis reveals whether the meta safety critic improves constraint satisfaction, fine-tuning comparison across baselines shows how CMAML balances adaptation speed and cost control, and CPO experiments demonstrate algorithm agnosticity.

5.5.2 Main Results

We evaluate CMAML across the 106-task distribution, examining both meta-training dynamics and fine-tuning performance. The results demonstrate that integrating safety constraints into meta-learning enables rapid adaptation while maintaining cost control, addressing the fundamental challenge of learning initializations that facilitate both efficient task completion and consistent constraint satisfaction.

Meta Safety Critic Effectiveness

The meta safety critic mechanism, regulated by the trainable Lagrange multiplier η , directly influences policy safety during both meta-training and subsequent fine-tuning. Figure 5.3 compares two variants of the framework: one employing an adaptive η guided by the safety critic, and another fixing $\eta = 0$ to eliminate the meta-level constraint.

During meta-training, the adaptive η maintains substantially lower cost margins throughout learning, avoiding the volatility observed when the meta-level constraint is absent. This suggests the safety critic successfully guides the outer loop optimization toward initializations that encode constraint-respecting behavior. Without this mechanism, the meta-learning process prioritizes rapid adaptation through return maximization, leading to initializations that discover high-reward strategies without internalizing safety considerations.

The effect persists into fine-tuning, where policies initialized with adaptive η demonstrate more consistent threshold adherence despite both approaches occasionally violating the $d = 5$ limit. This indicates that safety-aware meta-learning

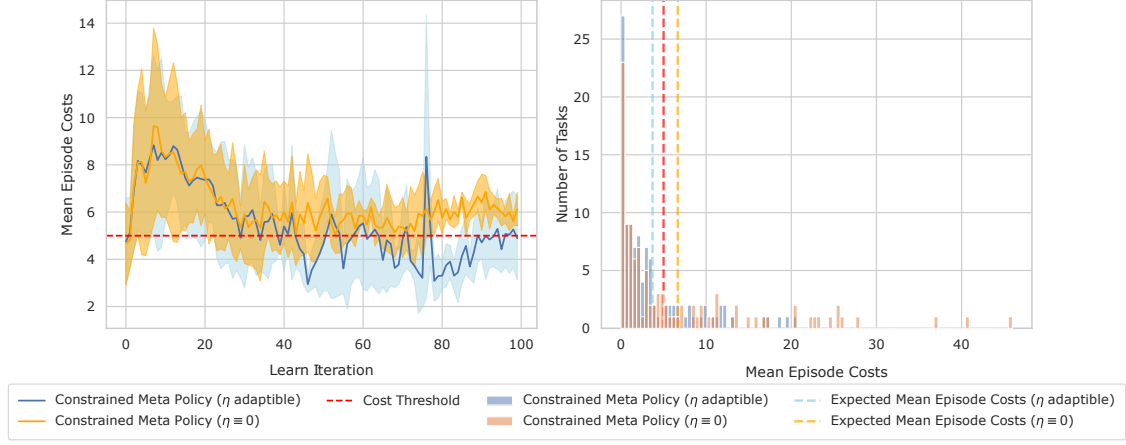


Figure 5.3: Impact of the meta safety critic on policy safety and adaptability during meta training. The adaptive safety weight η modulates the influence of the safety critic, encouraging exploration while preventing persistent violations. The resulting meta learned initialization yields policies that adapt quickly while keeping costs close to the $d = 5$ cost threshold when using the safety critic mechanism.

produces initializations requiring fewer fine-tuning steps to discover constraint-satisfying policies. The meta safety critic thus serves a dual purpose: regulating constraint satisfaction during meta-training while simultaneously shaping initializations that facilitate safer adaptation during deployment.

Adaptation Speed and Cost Control

Figure 5.4 presents fine-tuning performance across all four policy initializations, revealing how different meta-learning and pretraining strategies balance adaptation efficiency against constraint satisfaction.

CMAML achieves substantially faster adaptation than random initialization while maintaining episode costs close to the constraint limit throughout fine-tuning. The initialization encodes sufficient prior knowledge to accelerate learning without sacrificing safety, demonstrating successful integration of task performance and constraint satisfaction into the meta-learned parameters.

In contrast, both the pre-trained policy and unconstrained MAML achieve comparable or superior returns, but consistently violate cost constraints during early adaptation. The pretrained baseline demonstrates rapid return growth because it transfers task-solving knowledge from its training environment, but this knowledge lacks generalization of safety considerations across the task distribution. Similarly, unconstrained MAML learns an initialization that facilitates quick adaptation to novel obstacle configurations; however, this efficiency comes at the cost of discovering policies that prioritize reward maximization over constraint satisfaction.

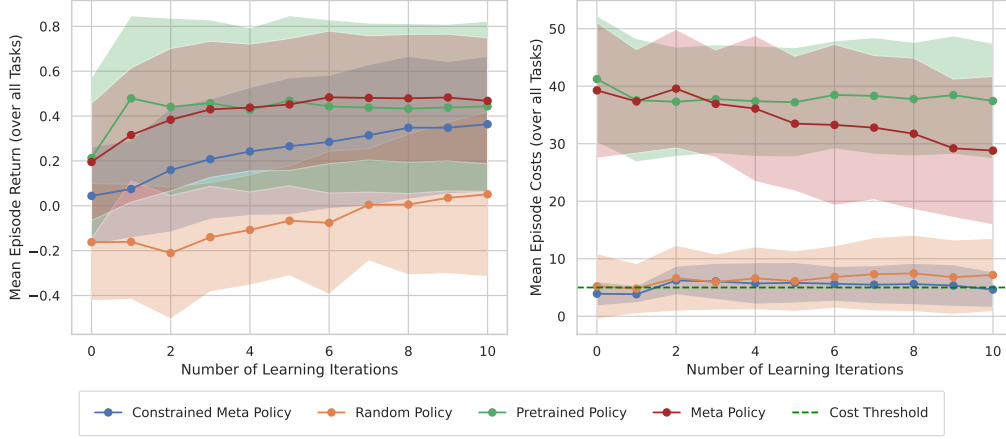


Figure 5.4: Mean episode return and cost during fine tuning across 106 tasks. CMAML with TRPOLag (blue) adapts efficiently while maintaining costs near the $d = 5$ threshold. Random initialization (orange) shows slow adaptation. TRPOLag pretrained (green) and unconstrained MAML (red) adapt quickly but incur substantially higher costs, revealing the safety cost of ignoring constraints during meta learning.

These results highlight a fundamental distinction between task-focused and safety-aware meta-learning. Learning to adapt quickly differs from learning to adapt safely. CMAML addresses this gap by encoding constraint satisfaction into the initialization itself, ensuring that rapid adaptation preserves safety rather than compromising it. The consistent performance across diverse obstacle configurations indicates that this safety-aware initialization generalizes effectively throughout the task distribution.

Algorithm Agnosticity

To verify that the framework’s benefits extend beyond specific algorithmic choices, we replaced TRPOLag with CPO in both the inner loop and fine-tuning phases. Figure 5.5 demonstrates that CMAML with CPO exhibits performance patterns consistent with the TRPOLag variant, maintaining the key characteristics of efficient adaptation coupled with cost control.

CMAML with CPO adheres to the relaxed $d = 10$ cost limit while demonstrating efficient adaptability across tasks. The similarity between Figures 5.4 and 5.5 indicates that the meta-learning framework successfully integrates different constrained policy optimization approaches without requiring algorithm-specific modifications. This confirms the model-agnostic principle underlying MAML extends naturally to constrained settings, where the choice of inner loop algorithm affects optimization mechanics without altering the fundamental meta-learning structure.

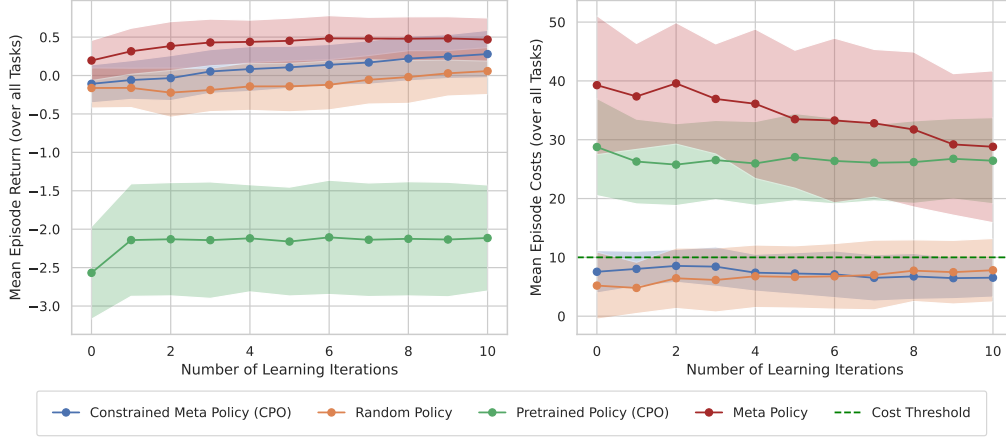


Figure 5.5: Fine tuning performance using CPO as the inner loop algorithm. CMAML with CPO (blue) maintains costs near the $d = 10$ threshold while adapting efficiently. Random initialization (orange), CPO pretrained (green), and unconstrained MAML with TRPO (red) exhibit similar patterns to those in Figure 5.4, confirming the framework generality across constrained optimization methods.

The consistent performance across both TRPOLag and CPO validates that safety-aware meta-learning does not depend on particular constraint-handling mechanisms. Instead, the framework provides a general approach for encoding safety considerations into meta-learned initializations, applicable to any constrained policy optimization method suitable for the inner loop adaptation process.

5.6 Discussion and Limitations

5.6.1 Safety–Adaptation Tradeoffs

The experimental results demonstrate that CMAML successfully addresses the tension between rapid task adaptation and constraint satisfaction during meta-learning. Compared to random initialization, the meta-learned parameters enable substantially faster adaptation across the 106 SafetyGymnasium navigation tasks while maintaining episode costs close to the prescribed thresholds throughout fine-tuning. This improvement stems from the dual-level constraint mechanism: task-specific constraints in the inner loop shape individual policy updates, while the outer-loop meta-constraint biases the initialization toward inherently safer regions of parameter space.

The comparison with unconstrained MAML and task-specific pretraining reveals the cost of ignoring safety during meta-learning. Both approaches achieve higher returns by violating safety constraints during adaptation, leveraging aggressive exploration that CMAML deliberately restricts. The pretrained policy, optimized

for a single task, generalizes poorly and exhibits pronounced cost spikes when adapting to different obstacle configurations. Unconstrained MAML prioritizes return maximization without accounting for constraint satisfaction during either meta-training or fine-tuning.

The meta safety critic and adaptive Lagrange multiplier η play complementary roles. The critic provides task-independent cost estimates under the meta-policy, enabling gradient updates that account for safety across the entire task distribution. The multiplier η regulates how strongly this safety signal influences meta-parameter updates, balancing exploration with constraint adherence. Fixing $\eta = 0$ removes this regularization, yielding more aggressive exploration but at the price of larger violations during both training and fine-tuning.

5.6.2 Model Agnosticism and Generalization

The successful integration of both TRPOLag and CPO as inner-loop optimizers validates the design principle of algorithm agnosticism. This flexibility matters because different constrained optimization methods offer distinct tradeoffs between computational efficiency, sample complexity, and constraint-satisfaction guarantees. The consistent performance across both methods indicates that the meta-level safety mechanisms (the distribution-level constraint and safety critic) operate largely independently of the specific inner-loop algorithm.

The experiments further illustrate how task-distribution characteristics influence effectiveness. Navigation environments with higher obstacle density create tasks in which collision avoidance becomes critical, whereas simpler layouts admit more direct paths to the goal. CMAML maintains consistent constraint satisfaction across these variations, although adaptation speed naturally decreases as task difficulty increases. The assumption of task-independent constraints (identical cost functions and thresholds across all tasks) is both enabling and limiting. It allows training a single meta safety critic from on-policy data gathered across tasks, which aligns well with robotic applications that share universal safety requirements (collision avoidance, joint limits, torque bounds). However, this formulation becomes less appropriate when different tasks demand fundamentally different safety priorities.

5.6.3 Limitations

The first-order meta-gradient approximation discards potentially valuable second-order information from the inner loop. Traditional MAML computes gradients through the entire adaptation trajectory, capturing how changes to the initialization propagate through multiple adaptation steps. CMAML approximates this dependency with the identity, treating inner-loop updates as fixed operators. The meta safety critic partially compensates by providing direct safety feedback in

the outer loop, but this addresses only safety and does not recover the broader adaptation information lost by omitting second-order terms.

The meta safety critic itself provides only approximate cost estimates, trained from on-policy rollouts that improve only as quickly as the meta-policy develops. Early in meta-training, the meta-policy explores broadly but generates low-quality data; later, as the policy converges, the critic sees higher-quality but less diverse trajectories. This creates a bootstrapping effect: the meta-policy needs reliable cost estimates to learn safely, while the critic needs a reasonably competent meta-policy to generate informative data. The experiments indicate that this interaction works sufficiently well in practice, but transient violations still occur, particularly during early meta-training.

Finally, the current evaluation focuses on continuous control tasks with low-dimensional state spaces. The SafetyGymnasium navigation tasks use compact state representations that standard neural networks can model effectively. Scaling CMAML to vision-based or contact-rich domains would likely require learned latent representations and more expressive function approximators, as well as more sophisticated uncertainty estimation in the presence of strong stochasticity. Moreover, the framework provides no formal safety guarantees during either meta-training or fine-tuning: constraints are handled through Lagrangian methods that guarantee only asymptotic satisfaction in expectation. Individual episodes may still violate constraints, and early training often exhibits more violations than later stages. For safety-critical applications where even occasional violations during learning are unacceptable, CMAML in its current form remains insufficient.

6 Scaling Off-Policy Learning for Practical Robotic Deployment

6.1 Overview

Deploying reinforcement learning on physical robots requires methods that learn quickly, run efficiently on edge hardware, and transfer reliably from simulation to reality. While previous chapters addressed safety through constrained optimization (Chapter 3) and rapid adaptation through meta-learning (Chapters 4–5), this chapter focuses on a complementary challenge: scaling off-policy learning to vision-based locomotion when computational resources are limited.

The dominant approach in legged robotics today relies on on-policy methods such as Proximal Policy Optimization (PPO) (Schulman et al. 2017), which achieve robust gaits by parallelizing training across thousands of simulated environments (Rudin et al. 2022, Makoviychuk et al. 2021). This strategy is effective when robots rely solely on proprioceptive feedback. Incorporating visual observations, however, introduces substantial complications. Visual encoders consume a significant amount of GPU memory, which limits the number of environments that can run simultaneously. Additionally, on-policy methods cannot reuse past experience, so every policy update requires new interactions. This inefficiency becomes particularly costly when training vision-based policies with limited parallelism or when fine-tuning policies on physical hardware, where each rollout is slow and physically expensive.

Off-policy algorithms provide a natural alternative through experience replay, enabling policies to learn from stored transitions and incorporate external datasets. Extending these methods to vision-based control, however, creates stability challenges. Fewer parallel environments mean less diverse training batches, which makes learning more brittle. Training at high update-to-data ratios amplifies this problem, as repeatedly processing the same visual observations causes critics to become overconfident about actions they have rarely seen (Nikishin et al. 2022). Recurrent architectures, which are necessary for partial observability, exacerbate these effects, since temporal correlations in the replay buffer cause errors to propagate over time (Hussing et al. 2024, Cetin et al. 2022).

This chapter presents a framework that addresses these challenges through architectural interventions and structured supervision rather than relying primarily on raw simulation throughput. Vision-based control inevitably reduces feasible

parallelism compared to proprioceptive settings, so stability must come from how the networks are parameterized and how supervision is applied. We adopt the SimbaV2 architecture (Lee et al. 2025b), which uses hyperspherical normalization to keep feature magnitudes bounded under non-stationary replay data. A recurrent encoder aggregates temporal context from egocentric depth sequences, allowing the policy to infer terrain properties that no single frame reveals. To counter the unreliable value estimates that plague off-policy learning from pixels, we introduce a hybrid training objective that combines reinforcement learning with time-decaying distillation from a privileged teacher. Strong supervision early in training prevents the actor from exploiting poorly calibrated critics. As the replay buffer grows and value estimates mature, the distillation weight decays and control transfers to the reinforcement learning objective. The framework builds on Fast Twin Delayed Deep Deterministic Policy Gradient (FastTD3) (Seo et al. 2025) and extends it with an asymmetric actor-critic formulation (Pinto et al. 2017): critics observe privileged terrain information during training, while the policy learns exclusively from deployable sensors. This asymmetry provides informative value targets during learning without compromising real-world deployment.

The chapter proceeds from related work (Section 6.2) and background (Section 6.3) through the complete framework (Section 6.4) to experimental validation in simulation and on hardware (Section 6.5), concluding with limitations and future directions (Section 6.6).

6.2 Related Work

6.2.1 Scaling Off-Policy Reinforcement Learning

Off-policy algorithms promise sample efficiency through experience replay, enabling learning from diverse data sources including demonstrations, prior deployments, and suboptimal exploratory trajectories. Realizing this potential at scale, however, introduces fundamental stability challenges. When update-to-data ratios increase beyond modest values, bootstrapped value estimates begin to diverge. Fujimoto et al. (2018) and Hussing et al. (2024) identify the core mechanism: early over-optimistic Q-values for poorly explored actions bias the policy toward those actions, causing their transitions to accumulate in the replay buffer and be repeatedly used to compute bootstrapped targets. This feedback loop amplifies early errors and creates a primacy bias, where initial experiences disproportionately shape learned values and reduce plasticity as learning progresses (Nikishin et al. 2022).

Architectural interventions address these pathologies through normalization schemes that decouple value scale from gradient scale (Lee et al. 2025a, Ball et al. 2023, Lee et al. 2025b), distributional value functions that predict return distributions rather than point estimates (Bellemare et al. 2017), and ensemble

critics that reduce overestimation by quantifying uncertainty (Chen et al. 2021b). Together, these techniques stabilize deep critics trained with noisy bootstrapped targets.

An alternative scaling strategy exploits computational throughput rather than architectural sophistication. Modern GPU simulators execute thousands of parallel environments and stream experiences directly to learners, converting hardware capacity into faster convergence (Makoviychuk et al. 2021). This regime favors large, diverse batches that provide stable gradients while keeping the effective update-to-data ratio moderate despite aggressive replay. Li et al. (2023) show that Parallel Q-Learning matches on-policy wall-clock performance in this setting while retaining superior sample efficiency. FastTD3 adopts a similar philosophy, coupling high throughput with lightweight architectural choices such as distributional critics, minimal normalization, and TD3’s clipped double-Q mechanism (Seo et al. 2025, Fujimoto et al. 2018).

These results are primarily demonstrated in settings with low-dimensional observations. In the vision-based locomotion problems considered in this chapter, visual encoders reduce feasible parallelism and change the balance between architectural interventions and throughput-based scaling. The next subsections review how these factors interact in quadrupedal control.

6.2.2 Vision-Based Locomotion under Partial Observability

Incorporating vision into locomotion control transforms the learning problem in several ways. Observations become high-dimensional: even a low-resolution depth image contains thousands of values, compared to the few dozen proprioceptive measurements typical for quadrupedal control. Egocentric views also provide only partial information about the surrounding terrain. Miki et al. (2022) and Agarwal et al. (2023) show that obstacle locations, surface geometry, and upcoming features must be inferred from temporal sequences rather than observed directly. In addition, the computational overhead of visual processing limits the parallel simulation throughput required for stable off-policy learning. Methods that work well with purely proprioceptive observations often become brittle when extended to pixels, especially with experience replay and high update frequencies (Cetin et al. 2022).

The partial observability inherent to egocentric vision requires policies to maintain and reason over temporal context. Recurrent architectures provide a natural mechanism for this integration. Miki et al. (2022) and Agarwal et al. (2023) use LSTM and GRU encoders to aggregate observation histories into latent representations that capture task-relevant properties. These encoders infer terrain geometry, surface characteristics, and obstacle configurations from egocentric depth and proprioceptive sequences. Temporal fusion enables anticipatory gait adjustments before physical contact occurs, which is essential for navigating rough terrain.

Simpler approaches trade expressiveness for efficiency: frame stacking concatenates fixed windows of recent observations into a single input tensor without a recurrent state (Mnih et al. 2015), while transformer architectures offer richer temporal modeling through self-attention (Vaswani et al. 2017, Chen et al. 2021a) at the cost of quadratic complexity that strains GPU memory budgets.

Visual processing directly constrains throughput. While Rudin et al. (2022) and Makovychuk et al. (2021) achieve 50,000 or more steps per second with proprioceptive policies using thousands of parallel environments, depth-based systems operate at substantially lower parallelism. Convolutional encoders and recurrent networks impose significant memory demands, and storing high-dimensional depth observations in the replay buffer further tightens the budget. Feasible batch sizes therefore shrink. This reduction matters: batch diversity influences gradient stability, and fewer environments lead to increased temporal correlations in replay buffers. Under these constraints, the architectural interventions discussed in Section 6.2.1 become crucial for maintaining stability at high update-to-data ratios.

6.2.3 **Asymmetric Critics and Teacher-Student Policies for Locomotion**

Asymmetric actor-critic architectures address partial observability by exploiting privileged information during learning while ensuring policies depend only on deployable sensors at execution time (Pinto et al. 2017, Kumar et al. 2021). Lee et al. (2020), Miki et al. (2022), and Agarwal et al. (2023) equip critics with ground-truth state, including detailed terrain information and contact signals. This privileged critic receives informative value targets even when egocentric observations are ambiguous. The policy, in contrast, processes only realistic inputs such as depth images and proprioceptive measurements. This asymmetry decouples the problem of learning effective behaviors from the problem of inferring state under partial observations. At deployment, only the policy is used and the critic is discarded.

Teacher-student formulations extend this idea by transferring behaviors from a privileged teacher to a deployable student. In the Reinforcement Learning with Prior Data (RLPD) framework (Ball et al. 2023), the teacher collects trajectories into an offline buffer, and an off-policy learner subsequently trains on a mixture of this buffer and online experience. The teacher supervision enters implicitly through the replay distribution and requires no change to the underlying reinforcement learning objective. In legged locomotion, Chane-Sane et al. (2024) follow a related philosophy by using privileged policies to generate expert data and then training a vision-based student on a combination of expert and on-policy rollouts.

Other systems perform teacher-student transfer explicitly through distillation losses. Miki et al. (2022) first train a privileged recurrent policy and then distill it into a vision-based controller by penalizing deviations between student and teacher actions, while maintaining the asymmetric critic during training. Agarwal

et al. (2023) apply a similar idea for outdoor deployment, combining privileged training with vision-based execution. High-agility systems from Hoeller et al. (2024) demonstrate parkour behaviors, including platform jumps and gap leaps, using depth perception and privileged training signals. Across these works, privileged information and teacher supervision provide stable learning signals, while deployed policies master the perceptual inference required for real-world operation. In this chapter, we adopt this teacher-student perspective and study how time-decaying supervision interacts with throughput-scaled off-policy learning and recurrent vision encoders.

6.3 Background and Problem Formulation

This section establishes the technical foundations for scaling off-policy vision-based locomotion. We build on the off-policy learning framework from Section 2.1.3 and the partial observability setting from Section 2.1.4. Within this context, we review Twin Delayed Deep Deterministic Policy Gradient (TD3) and its throughput-scaled extension FastTD3, which serves as the base algorithm in this chapter. The section concludes with the teacher-student learning paradigm, which we later combine with FastTD3 and our recurrent vision architecture.

6.3.1 Twin Delayed Deep Deterministic Policy Gradient

Within the off-policy framework from Section 2.1.3, TD3 (Fujimoto et al. 2018) addresses the overestimation bias that destabilizes actor-critic methods trained from replay buffers. The core problem is that function approximation errors compound through bootstrapping: the critic learns from its own imperfect predictions, and the actor exploits these errors by selecting actions that appear valuable but are in fact overestimated.

TD3 reduces this bias through three design choices: clipped double Q-learning, delayed policy updates, and target policy smoothing. Clipped double Q-learning maintains two critics and uses their minimum when computing targets:

$$y = r + \gamma \min_{i=1,2} Q_{\psi'_i}(s', \pi_{\theta'}(s')). \quad (6.1)$$

Here, ψ'_i and θ' denote the parameters of the target critics and target policy, and taking the minimum yields a conservative estimate that counteracts systematic overestimation. Delayed policy updates further improve stability by letting the critics adapt before the actor is updated; in practice, we update the policy once for every two critic updates. Target policy smoothing then adds clipped noise to target actions, which prevents the critic from developing sharp, narrow peaks that the actor could exploit.

Together, these mechanisms enable stable learning from diverse replay data and make TD3 a strong foundation for scalable off-policy algorithms. FastTD3 builds directly on TD3 and augments it with throughput-based scaling and distributional critics, as summarized in the next subsection.

6.3.2 FastTD3: Throughput-Based Scaling

While TD3 stabilizes off-policy learning, high update-to-data ratios amplify bootstrapping errors when the number of gradient updates far exceeds the number of fresh samples. FastTD3 (Seo et al. 2025) addresses this by extending TD3 with massively parallel simulation, large-batch updates, and distributional critics.

The key idea is straightforward: generating samples fast enough keeps the effective update-to-data ratio low. GPU-accelerated simulators such as IsaacLab (Mittal et al. 2023) run thousands of environments in parallel and produce orders of magnitude more transitions per wall-clock second than traditional setups. This abundance of fresh data continuously refreshes the replay buffer and reduces the distribution shift between the behavior policy and the current policy.

Large batch sizes complement this high throughput. With batches of up to 32,000 transitions, each gradient step covers diverse regions of the state-action space and yields stable learning signals even without architectural mechanisms such as layer normalization or residual connections. Stability is further improved by a distributional critic, which learns the full return distribution rather than scalar expectations, provides smoother gradients, and prevents overconfident value estimates on out-of-distribution actions. Section 6.3.3 briefly reviews this formulation.

Together, these modifications allow FastTD3 to solve complex locomotion tasks in hours rather than days while maintaining training stability. In this chapter, FastTD3 serves as the base off-policy algorithm. Section 6.4 builds on this foundation and introduces recurrent vision encoders and a teacher-student objective for vision-based control under partial observability.

6.3.3 Distributional Reinforcement Learning

Standard critics predict a single scalar value $Q^\pi(s, a)$ that represents the expected return. This scalar view discards information about how returns may vary across rollouts. Distributional RL (Bellemare et al. 2017) addresses this limitation by modeling the full return distribution $Z^\pi(s, a)$ instead of only its expectation. The distributional Bellman equation characterizes how return distributions propagate:

$$Z^\pi(s, a) \stackrel{d}{=} r(s, a) + \gamma Z^\pi(s', a'), \quad (6.2)$$

where $\stackrel{d}{=}$ denotes equality in distribution rather than equality of scalar values.

The categorical representation, introduced in C51 (Bellemare et al. 2017), approximates $Z^\pi(s, a)$ as a discrete distribution over a fixed support $\{z_1, \dots, z_N\}$ spanning the range of possible returns. The critic outputs a probability vector $p_\theta(s, a) = (p_1, \dots, p_N)$ over these atoms, and training minimizes the cross-entropy between the projected Bellman target and the current estimate:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[\sum_{i=1}^N p_{\text{target},i} \log p_{\theta,i}(s, a) \right], \quad (6.3)$$

where p_{target} is obtained by applying the distributional Bellman operator and projecting the resulting distribution back onto the fixed support.

This distributional perspective provides two benefits that are important for scaling off-policy learning. First, predicting full distributions yields smoother gradients than scalar regression, as the cross-entropy loss provides informative learning signals even when target distributions shift gradually. Second, the distribution’s shape conveys uncertainty: wide distributions indicate high-variance returns, while narrow distributions suggest predictable outcomes. For FastTD3, these properties stabilize training at high update-to-data ratios by preventing the overconfident value estimates that typically cause divergence when critics train repeatedly on limited data.

6.3.4 Teacher-Student Learning and Knowledge Distillation

Section 6.2.3 reviewed how teacher-student learning transfers behaviors from privileged teachers to deployable students. Teacher supervision can enter the learning process in two main ways. One option is to keep the reinforcement learning objective unchanged and populate the replay buffer with trajectories generated by the teacher, so supervision enters implicitly through the data distribution. A second option is to replace reinforcement learning with supervised imitation and train the student solely to match the teacher:

$$\mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}_{s \sim \mathcal{D}} [\|\pi_\theta(s) - \pi_{\text{teacher}}(s)\|^2], \quad (6.4)$$

where π_θ is the student policy and π_{teacher} provides reference actions for states sampled from the teacher dataset $\mathcal{D}$. We use both variants as baselines to assess how different forms of teacher supervision affect training stability and control performance. Section 6.4.2 builds on this formulation and introduces a time-decaying distillation term that is combined with the actor’s reinforcement learning objective.

6.3.5 Problem Formulation

The learning problem is to train a quadruped locomotion policy that uses ego-centric depth images and proprioceptive measurements to traverse varied terrain.

During training, the critic observes privileged terrain state while the actor sees only onboard sensors. At deployment, only the actor is used. The challenge is obtaining a vision-based policy that benefits from off-policy sample efficiency while remaining stable under partial observability and experience replay.

6.4 Method

Training vision-based locomotion policies with off-policy methods raises two challenges that interact. Single depth frames cannot fully characterize terrain geometry, leaving the policy unable to anticipate obstacles beyond the current field of view. At the same time, critics provide unreliable value estimates early in training when experience is limited, which encourages the actor to exploit poorly calibrated predictions.

We address these issues with two complementary mechanisms. A recurrent architecture aggregates temporal context from observation histories, allowing the policy to infer terrain properties that no single frame can reveal. A hybrid teacher-student objective supplies behavioral supervision while the critic is still immature and gradually decays as value estimates become more reliable.

The following subsections describe the network architecture that supports stable learning at high update-to-data ratios, the hybrid training objective and its decay schedule, and the action space representation and control interface used for deployment.

6.4.1 Network Architecture

Scaling off-policy learning to high update-to-data ratios introduces the stability challenges discussed in Section 6.2.1. We address these through two mechanisms: hyperspherical normalization to control feature magnitudes under non-stationary replay, and a recurrent encoder to aggregate temporal context from egocentric depth sequences.

Both actor and critic use SimbaV2 networks (Lee et al. 2025b), which constrain weights and hidden activations to lie on a hypersphere. This normalization stabilizes optimization under non-stationary replay data and reduces reliance on weight decay or periodic reinitialization.

The actor-critic setup follows the asymmetric formulation from Section 2.1.4: critics receive privileged terrain observations that fully characterize local geometry, while the actor processes only deployable inputs, specifically egocentric depth images and proprioceptive measurements. This separation enables the critic to exploit rich state information to shape gradients while maintaining the learned policy’s executability on the real robot.

The actor processes observations through a three-stage pipeline. A compact convolutional encoder maps depth images to a low-dimensional latent vector. During training, we apply random spatial shifts, additive noise, and cutout to improve robustness to viewpoint variation and sensor noise (Appendix D.1). We then concatenate the latent vector with proprioceptive observations and feed the result to a GRU that aggregates temporal context. The recurrent state helps infer terrain properties and obstacle configurations from observation histories, compensating for ambiguities that single frames cannot resolve. Finally, the GRU output passes through the SimbaV2 policy network to produce actions.

We keep the encoder deliberately lightweight to preserve GPU memory for many parallel environments. This limited visual capacity maintains high simulator throughput and batch diversity, both of which are essential for stable learning at aggressive update-to-data ratios. The replay buffer stores fixed-length sequences and applies gradient masking after terminal transitions to prevent backpropagation across trajectory boundaries (Appendix D.2).

On the value side, critics use the distributional formulation from Section 6.3.3, predicting return distributions over a fixed categorical support rather than scalar expectations. On the policy side, actors combine off-policy reinforcement learning with time-decaying behavioral supervision from the teacher, as detailed in Section 6.4.2.

6.4.2 Hybrid Teacher-Student Training with Time-Decaying Knowledge Distillation

Off-policy learning from visual observations introduces challenges that architectural and throughput-based solutions alone cannot resolve. Pixel inputs create high-dimensional, partially observable state representations where critics struggle to form reliable value estimates during early training. We address this through behavioral guidance from a privileged teacher that decays as the student’s estimates mature, as illustrated in Figure 6.1.

Training proceeds in two phases. In the first phase, the teacher policy π_{teacher} learns using PPO on terrain height scans and roof clearance measurements that are unavailable to physical sensors. These observations fully characterize local obstacle geometry and enable effective locomotion strategies that serve as a behavioral reference for subsequent learning.

Once the teacher converges, student policy π_{θ} trains on egocentric depth and proprioception, shaped by two supervision signals. The first is the RL loss that maximizes expected Q-value under the learned critic:

$$\mathcal{L}_{\text{RL}}(\theta) = -\mathbb{E}_{(o, s^{\text{priv}}) \sim \mathcal{D}} \left[Q_{\psi}(o, s^{\text{priv}}, \pi_{\theta}(o, h_{t-1})) \right], \quad (6.5)$$

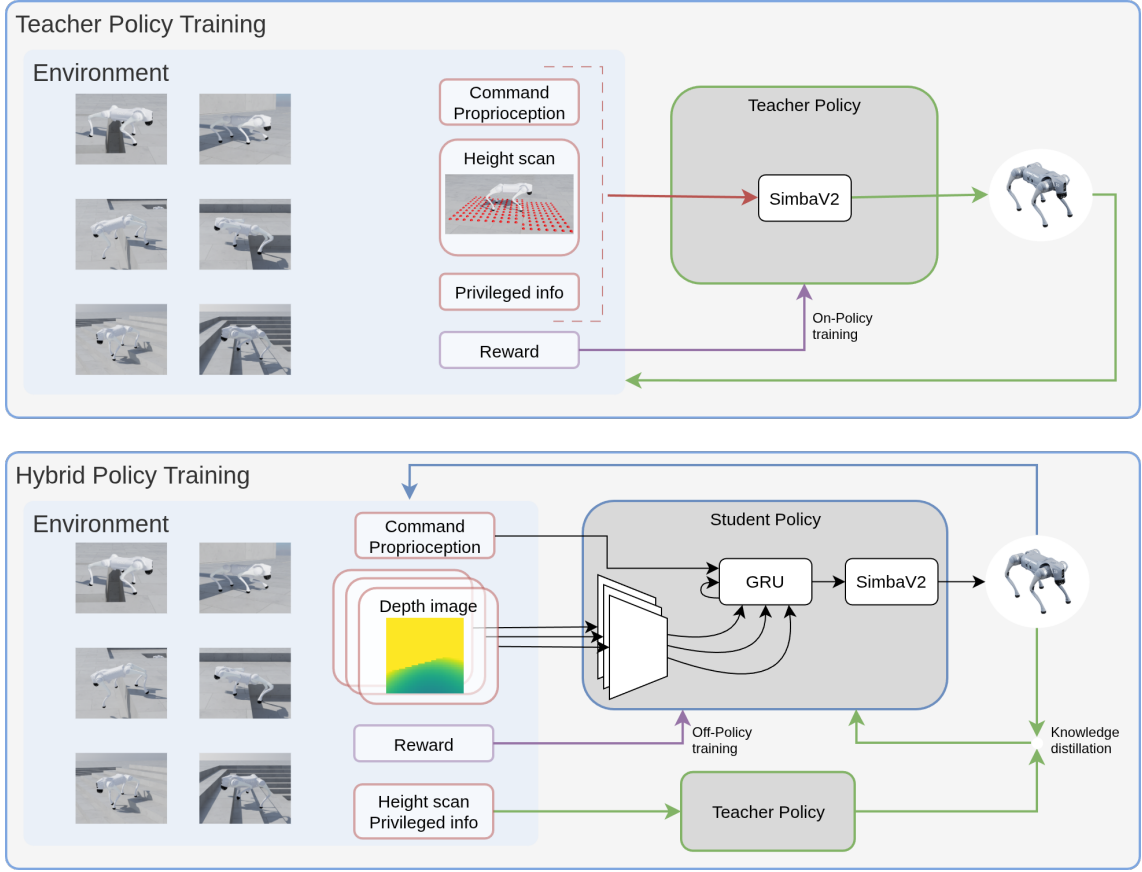


Figure 6.1: Two-phase training architecture. Top: teacher policy training with privileged terrain information using on-policy PPO. Bottom: student policy training with dual supervision from off-policy TD3 updates and time-decaying knowledge distillation from the teacher. The student processes egocentric depth images through a CNN encoder and GRU, while the teacher runs in parallel providing behavioral targets based on privileged geometric observations.

where o denotes student observations, s^{priv} are privileged states provided to the critic, and h_{t-1} is the recurrent hidden state. The second is knowledge distillation, which penalizes deviations from teacher actions:

$$\mathcal{L}_{\text{KD}}(\theta) = \mathbb{E}_{(o, s^{\text{priv}}) \sim \mathcal{D}} \left[\left\| \pi_{\theta}(o, h_{t-1}) - \pi_{\text{teacher}}(s^{\text{priv}}) \right\|_2^2 \right], \quad (6.6)$$

where teacher policy π_{teacher} runs in parallel, continuously providing target actions from privileged observations.

The complete objective combines both signals:

$$\mathcal{L}_{\pi}(\theta) = \mathcal{L}_{\text{RL}}(\theta) + \lambda(t) \cdot \mathcal{L}_{\text{KD}}(\theta), \quad (6.7)$$

with knowledge distillation weight decaying exponentially:

$$\lambda(t) = \lambda_{\text{max}} \cdot e^{-\alpha t/T}. \quad (6.8)$$

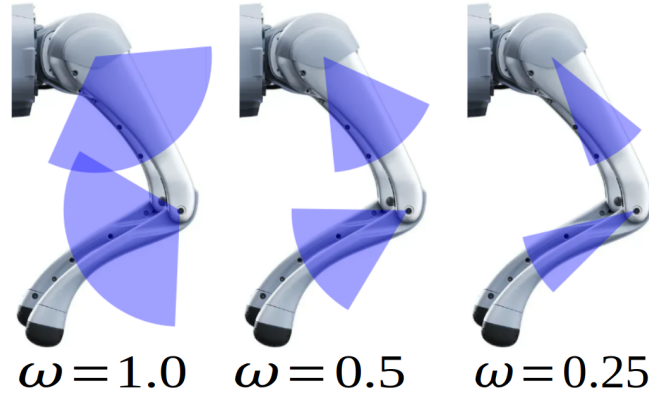


Figure 6.2: Effect of action scaling parameter ω on effective joint workspace for a single GO2 leg. Larger values (left, $\omega = 1.0$) allow large joint excursions for a given change in policy output, while smaller values (middle, $\omega = 0.5$; right, $\omega = 0.25$) yield more conservative targets. We use $\omega = 0.25$ to balance stability with fast corrective motions.

Here t is the current training step, T is total steps, $\lambda_{\max}$ controls initial supervision strength, and α determines decay rate. Strong supervision early in training stabilizes exploration while the critic matures; as the replay buffer grows and value estimates improve, the decaying weight transfers control to reinforcement learning.

6.4.3 Action Space and Control Interface

Beyond network design and training objectives, action space parameterization and its interface to the low-level controller also influence stability and performance. The policy outputs joint position targets through a Proportional Derivative (PD) controller, represented as offsets from a default standing pose $\hat{q}$:

$$q_{\text{target}} = a \cdot \omega + \hat{q}, \quad (6.9)$$

where $a \in \mathbb{R}^{12}$ are raw policy outputs and ω is the action scaling parameter.

Unlike FastTD3, we remove the hyperbolic tangent activation from the actor output layer, letting the policy produce unbounded actions limited only by subsequent clipping. This avoids additional squashing nonlinearity and allows the policy to exploit the full mechanical range of joints up to safety limits. Scaling factor ω controls how aggressively changes in policy output translate into joint targets: larger values produce larger joint excursions for the same change in a , while smaller values yield more conservative motions. Figure 6.2 illustrates how different choices of ω affect the effective workspace of a single GO2 leg. We set $\omega = 0.25$ as a compromise between stability and the ability to generate fast corrective motions.

Commanded targets are clipped to actuator-safe limits before execution:

$$q_{\text{safe}} = \text{clip}(q_{\text{target}}, q_{\text{min}}, q_{\text{max}}), \quad (6.10)$$

where q_{min} and q_{max} denote joint limits. A PD controller tracks these clipped targets:

$$\tau = K_p(q_{\text{safe}} - q) + K_d(\dot{q}_{\text{safe}} - \dot{q}), \quad (6.11)$$

with proportional gains $K_p = 20$ and derivative gains $K_d = 0.5$ tuned to Unitree GO2 dynamics. Control runs at 50 Hz, providing smooth and responsive motion.

6.5 Experimental Evaluation

We evaluate the proposed framework on challenging quadrupedal locomotion tasks requiring vision-based terrain perception and agile motor control. The experimental validation addresses three key questions:

1. How do different teacher-student training strategies compare in terms of final performance and training stability when extending off-policy methods to high-dimensional vision inputs?
2. What is the impact of the behavior cloning weight and decay schedule on the balance between off-policy learning and imitation in the hybrid approach?
3. Does the learned vision-based policy successfully transfer to real-world deployment, and what behaviors emerge on the physical robot?

6.5.1 Experimental Setup

All experiments are conducted using IsaacLab, a GPU-accelerated physics simulation framework built on NVIDIA Isaac Sim (Makoviychuk et al. 2021), which enables the massively parallel environment execution necessary for high-throughput off-policy learning. We construct six distinct terrain scenarios representing different locomotion challenges: rough heightfields, geometric discontinuities, overhead constraints, and multi-level climbing tasks. This diversity allows us to assess whether vision-based policies generalize across fundamentally different obstacle types or overfit to specific geometric patterns. Following simulation training, we deploy the learned policies on a physical Unitree GO2 quadruped robot to evaluate sim-to-real transfer.

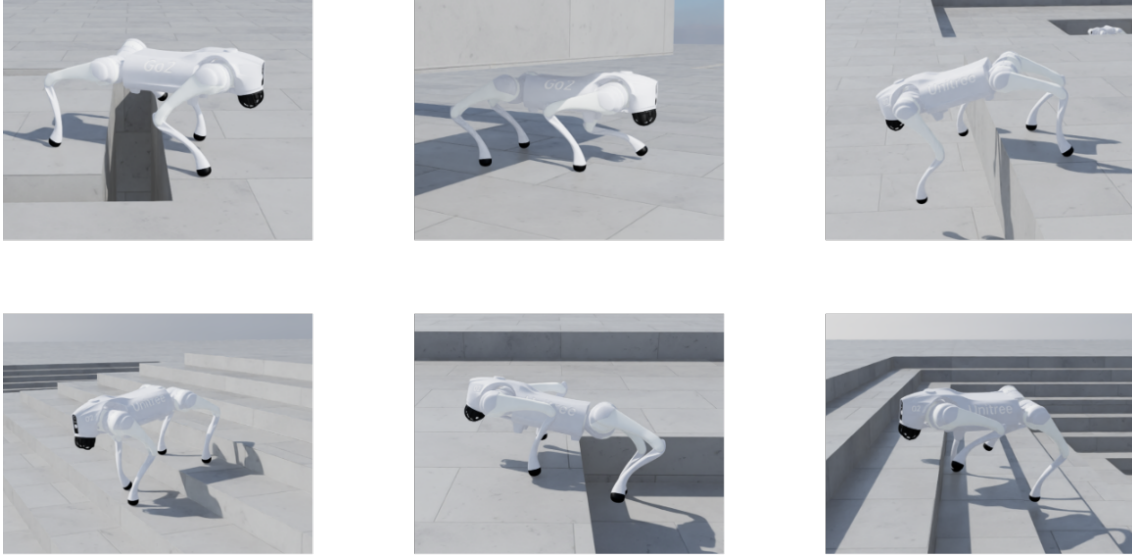


Figure 6.3: Training scenarios testing different locomotion capabilities. Top row (left to right): gap crossing requires precise foot placement, ring navigation demands dynamic crouching, box descent challenges controlled lowering. Bottom row (left to right): inverted pyramid stairs, pit recovery, and normal pyramid stairs. Pit recovery tests climbing power, while both pyramid terrains require adaptation to variable step geometry. All scenarios run across 512 parallel environments with randomized parameters.

Task Environments

Figure 6.3 illustrates the terrain scenarios. Gap crossing places the robot on a raised platform separated from the main floor by a 0.3 m discontinuity, while ring navigation suspends an obstacle at chest height that forces the robot to lower its body. Box descent arranges stacked boxes into a staircase with 0.15–0.25 m step heights. The remaining scenarios test vertical agility: pit recovery creates a two-tier pit demanding strong upward motion, and pyramid stairs form variable-geometry steps for ascending and descending.

Each scenario implements a progressive curriculum, tracked independently within each environment. After every episode, the curriculum evaluates the planar distance from the robot base to the environment origin. If this distance exceeds half the terrain length, the environment advances to a more challenging patch; if the robot covers less than half the distance implied by its commanded velocity, the environment reverts to an easier patch.

Domain randomization diversifies training conditions across several axes. At startup, we sample friction coefficients uniformly from $[0.3, 1.2]$ and restitution from $[0, 0.15]$ for all contact surfaces, and perturb base mass by a uniform offset in $[-1, 3]$ kg. During episodes, the base experiences periodic horizontal velocity pushes with components sampled from $[-0.5, 0.5]$ m/s. Episodes terminate when

the time horizon of 20 s is reached, when the base link contacts the environment, or when body orientation exceeds 0.8 rad (approximately 46°) in roll or pitch.

Robot Platform

The Unitree GO2 is a 12-Degree of Freedom (DOF) quadruped with three actuated joints per leg. Position control runs at 50 Hz with PD gains $K_p = 20$ and $K_d = 0.5$, matched between simulation and physical deployment.

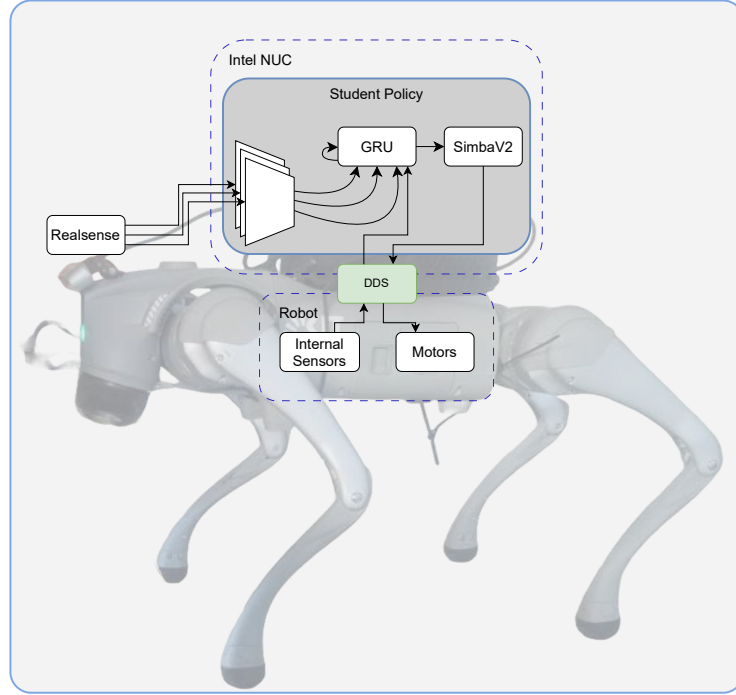


Figure 6.4: Deployment system architecture. The student policy runs on an Intel NUC payload, processing depth images from a RealSense camera and proprioceptive measurements received via DDS communication with the onboard computer. Joint position commands are sent at 50 Hz.

Figure 6.4 shows the deployment architecture. The trained student policy runs on an Intel NUC (11th Gen i7, 16 GB RAM) mounted as payload. We export the network to ONNX format for efficient CPU inference, achieving approximately 20 Hz policy execution. A RealSense D435i depth camera provides egocentric observations at 30 Hz, with images downsampled to 48×48 resolution and depth values thresholded at 1.0 m to match simulation preprocessing.

Baseline Methods

We compare five approaches that differ only in how teacher information guides student learning.

PPO Teacher. An on-policy PPO agent trained on privileged observations including height and roof scans. This baseline establishes the performance ceiling achievable with full state information and serves as the reference for teacher-student methods.

FastTD3 Baseline. FastTD3 (Seo et al. 2025) applied directly to vision-based locomotion without teacher supervision. The critic observes privileged information following asymmetric training (Pinto et al. 2017), but no demonstrations guide the actor.

RLPD. The reinforcement learning with prior data framework (Ball et al. 2023) combines online student rollouts with a fixed offline buffer containing 2,048,000 teacher transitions. Mini-batches sample equally from both buffers throughout training.

Knowledge Distillation. The teacher runs in parallel during training and provides action targets through a constant-weight supervised loss. The student learns from imitation.

Hybrid. Our approach extends off-policy RL with time-decaying supervision. Strong teacher guidance during early training stabilizes learning while the critic matures, then gradually fades as value estimates become reliable.

All vision-based methods share identical architectures: a convolutional encoder for depth, a GRU for temporal fusion, and SimbaV2 (Lee et al. 2025b) for actor and critic networks. Appendix D.3 details the complete specifications.

Evaluation Protocol

Performance Metrics. Terrain level achieved during training reflects curriculum progression, with each level requiring traversal of increasingly challenging obstacles. Mean episodic reward captures behavioral quality within each level. We conduct five independent runs per configuration with different random seeds and report learning curves showing mean and standard deviation.

Training Dynamics. For the hybrid approach, we analyze how $\lambda_{\max}$ and decay rate α influence optimization.

Sim-to-Real Transfer. Physical deployment assesses whether learned behaviors generalize beyond simulation, focusing on crawling under obstacles and flat-ground stability.

6.5.2 Overall Comparison of Approaches

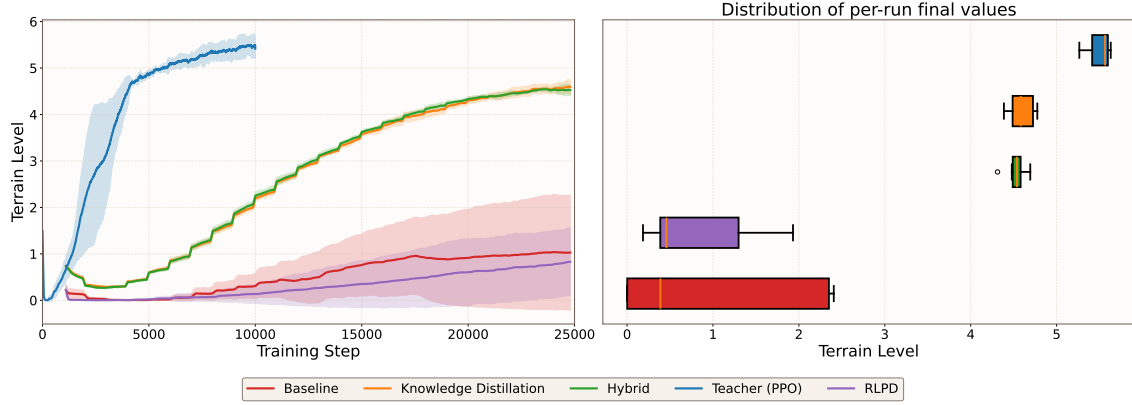


Figure 6.5: Terrain level performance across training for all evaluated approaches. The teacher (privileged PPO) reaches the highest levels, while knowledge distillation and the hybrid method perform similarly and substantially better than the baseline and RLPD. Right panel shows distribution of final terrain levels across runs.

Figure 6.5 shows terrain level progression across all evaluated approaches. The teacher agent reaches the highest terrain levels, which follows from its access to privileged state information unavailable to student policies.

Among student methods, knowledge distillation and hybrid perform almost identically in terms of terrain level, with both progressing through approximately four levels. This similar performance suggests that knowledge distillation already extracts most of the information recoverable from partial observations. The partial observability of the environment likely limits achievable performance for both methods.

The FastTD3 baseline, which learns purely from its own experience without teacher guidance, and RLPD, which mixes an offline teacher buffer with online student data, perform substantially worse and do not make meaningful terrain progress. For FastTD3, this is consistent with its design: the original method relies on thousands of parallel environments and huge batch sizes to keep the effective update-to-data ratio moderate, whereas our vision-based setting uses fewer environments, smaller batches, and a higher-dimensional, partially observable observation space, so its throughput advantage largely disappears and the critic becomes harder to stabilize. In RLPD, each mini-batch consists of 50% samples from a fixed teacher buffer and 50% from the student’s growing replay buffer. The teacher dataset is small relative to the student buffer and collected under a different training regime, so its state-action distribution only partially overlaps with the student’s recurrent, vision-based policy. As training progresses, these off-distribution teacher samples are replayed repeatedly, and their influence becomes highly sensitive to the particular trajectories stored. In our experiments, this offline use of teacher data did not provide a stable benefit. In contrast, methods with explicit, online action-level

distillation (knowledge distillation and hybrid) transferred sensorimotor behavior more reliably from privileged teachers to vision-based students.



Figure 6.6: Mean reward comparison across hybrid training variants with different knowledge distillation weights and decay schedules. Left: mean reward over training. Right: distribution of final mean rewards across runs.

Figure 6.6 shows the effect of knowledge distillation weight and decay on mean reward during hybrid training. The variant with $\lambda_{\max} = 0.01$ achieves lower final rewards than the variants with larger distillation weights.

Effect of Knowledge Distillation Weight

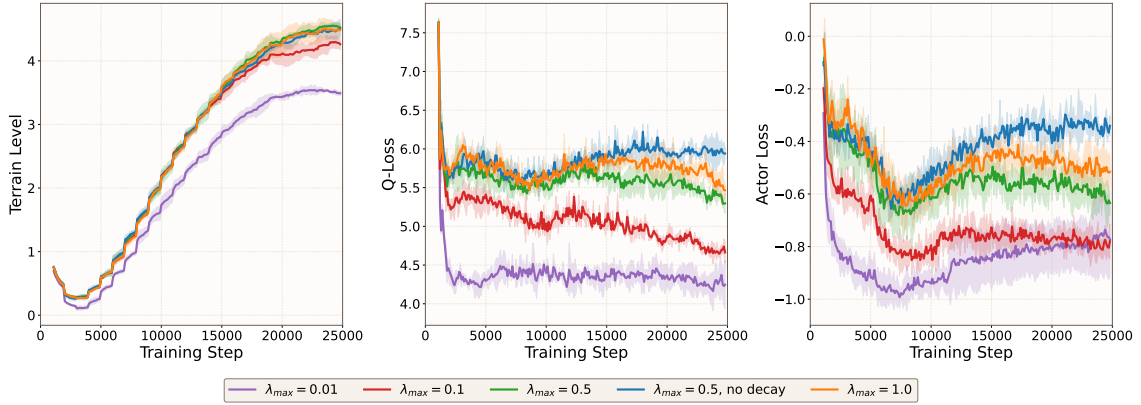


Figure 6.7: Effect of knowledge distillation weight $\lambda_{\max}$ and decay α on hybrid training. Top: terrain level (left), critic loss (middle), and actor loss (right) over training. For $\alpha > 0$, the distillation weight $\lambda(t)$ starts at $\lambda_{\max}$ and decays exponentially; for $\alpha = 0$, it remains constant.

Figure 6.7 illustrates how the trade-off between reinforcement learning and knowledge distillation depends on $\lambda_{\max}$ and the decay rate α . When $\lambda_{\max}$ is very small ($\lambda_{\max} = 0.01$), the distillation term is almost negligible at every point in training,

so the hybrid objective behaves essentially like pure off-policy RL. This yields the lowest critic and actor losses but also the lowest terrain level: without a meaningful teacher signal, the policy struggles to discover strong behaviors from vision alone. For moderate initial weights ($\lambda_{\max} \in \{0.1, 0.5, 1.0\}$ with $\alpha = 3$), the teacher term is substantial early on and then decays, which slightly worsens the losses but enables the agent to reach higher terrain levels. In this regime, strong early supervision stabilizes exploration and helps the critic focus on relevant parts of the state space, while the decay later allows the RL term to refine the policy beyond pure imitation.

The comparison between $\lambda_{\max} = 0.5$ with and without decay highlights the role of α . When $\alpha = 0$, the distillation weight stays at 0.5 throughout training, so the policy is continuously pulled toward the teacher and the RL term can only make small corrections. This configuration reaches comparable terrain levels but exhibits the worst critic and actor losses and higher variance, indicating a more brittle optimization process. With decay ($\alpha = 3$), the same initial weight provides strong guidance only when the critic is still immature and then gradually hands over control to the RL objective. This pattern matches observations in offline-to-online RL, where behavior-cloning regularization improves stability at the beginning but must be reduced over time to avoid overly conservative policies and slow improvement.

6.5.3 Performance on the Real Robot



Figure 6.8: Geometric setup for the crawling task. The robot is 45 cm tall with the payload, while the cardboard obstacle provides only 27 cm clearance, so a substantial posture change is required to pass underneath.

In real-world trials, the policy detected low obstacles and executed crawling motions underneath them. Figure 6.8 shows the geometric challenge: the robot stands 45 cm tall with a payload, while the obstacle provides only 27 cm clearance.



Figure 6.9: Sequence of the robot approaching the obstacle, lowering its body, and crawling underneath it during real-world deployment.

Figure 6.9 illustrates the resulting behavior. The robot lowers its body posture and passes under the obstacle, demonstrating successful transfer of the crawling skill from simulation to hardware.

The policy consistently initiates body lowering when depth images reveal overhead surfaces within approximately 0.5 m. The robot maintains stable locomotion throughout the transition between normal walking and crawling postures, indicating that temporal fusion enables smooth behavioral adaptation despite significant kinematic changes. At the same time, a residual gap remains between simulated and real-world robustness.

Several factors likely contribute to this sim-to-real gap. Domain randomization during training may have been insufficient for capturing real-world visual variation, particularly in lighting conditions and surface textures that are absent from simulation. The relatively small number of parallel environments (512 compared to 4096 for proprioceptive training) limited visual diversity. The lightweight encoder architecture, chosen to preserve GPU memory for parallelization, may also lack capacity for robust terrain feature extraction.

Despite these limitations, the successful transfer of crawling behavior shows that off-policy learning can produce vision-based policies that function on physical hardware. The hybrid training objective provided sufficient stability to learn meaningful visual representations, and the recurrent architecture successfully aggregated temporal context across real sensor streams with their attendant noise and latency.

6.6 Discussion and Limitations

6.6.1 Teacher Supervision and Off-Policy Integration

The results reveal a clear divide between how teacher knowledge can be incorporated into off-policy learning. Online, action-level supervision through knowledge distillation and the hybrid objective transfers sensorimotor skills reliably from privileged teachers to vision-based students. Offline replay of teacher transitions through RLPD, by contrast, proves brittle and sensitive to the particular trajectories stored. This gap reflects a fundamental challenge: when the teacher’s state-action distribution diverges substantially from the student’s evolving recurrent policy, replaying fixed offline data introduces distributional mismatch that destabilizes critic learning rather than accelerating it.

The hybrid approach offers a practical resolution to this tension. Strong supervision early in training prevents the actor from exploiting poorly calibrated critics, while the subsequent decay allows the reinforcement learning objective to refine behavior beyond what imitation alone achieves. The reward improvements over pure knowledge distillation confirm that genuine off-policy learning takes place alongside the supervised component, and not merely that the teacher signal dominates the update. At the same time, the sensitivity to $\lambda_{\max}$ and α shows that balancing the two objectives remains largely empirical. A principled scheduling mechanism, for instance one that conditions the distillation weight on critic convergence or value estimation quality, would reduce the need for manual hyperparameter sweeps.

6.6.2 Vision Representations and Partial Observability

In real-world deployment, the policy reliably detects low overhead obstacles and lowers its body but does not appear to use visual input for other obstacle types such as step-ups or irregular terrain. For those scenarios, the robot instead relies on proprioceptive and tactile feedback, a strategy that transfers poorly to hardware because the resulting contact-heavy motions are too aggressive for stable locomotion.

This specialization likely reflects an interaction between the curriculum structure and the reward signal. Crawling requires an unambiguous visual cue, a low overhead surface, that differs qualitatively from flat-ground walking. Other obstacle types can be partially solved through proprioceptive adaptation alone, which provides insufficient gradient signal for the encoder to learn discriminative visual features. Addressing this limitation would require either architectural modifications that strengthen visual representation learning, such as contrastive pretraining or auxiliary reconstruction objectives, or curriculum changes that more explicitly reward vision-dependent behaviors across all terrain types.

6.6.3 Limitations

Several factors constrain the scope of the presented results. The sim-to-real transfer, while successful for crawling, remains incomplete for other obstacle types. Domain randomization during training was limited by computational budget, and the relatively small number of parallel environments (512 compared to 4096 in proprioceptive settings) restricted the diversity of visual conditions encountered during learning. The lightweight encoder architecture, chosen to preserve GPU memory for parallelization, may lack sufficient capacity for robust feature extraction across the varied lighting and surface conditions present in real deployment.

The experimental evaluation is also restricted to a single robot platform and a single task domain. Whether the hybrid training strategy generalizes to different morphologies, alternative sensor modalities, or tasks with substantially different temporal structure remains an open question. The reliance on a privileged teacher further imposes a practical requirement: the entire pipeline assumes access to a high-performing policy trained on full state information, which may not always be straightforward to obtain for more complex environments.

Finally, the off-policy component operates with a deterministic policy, which limits exploration and prevents the use of policy entropy as a signal for managing the balance between imitation and reinforcement learning. Extending the framework to stochastic policies could enable more principled scheduling of the distillation weight, for instance by increasing teacher guidance when the policy’s action variance is high and reducing it as the student becomes confident in its own decisions.

7 Conclusion

7.1 Summary of Contributions

Deploying reinforcement learning on physical robots requires solutions to three interconnected challenges: learning must proceed safely without catastrophic failures, efficiently without requiring prohibitively large amounts of real-world data, and adaptively without losing effectiveness when conditions change. This dissertation has developed algorithmic frameworks that address these requirements through complementary technical approaches, progressing from foundational constrained optimization through meta-learning and culminating in practical deployment systems. This chapter summarizes the contributions, examines their interconnections, and outlines directions for future research.

7.1.1 Constrained Model-Based Policy Optimization

Can a robot learn to be safe by practicing in its imagination, even when that imagination is imperfect?

Chapter 3 introduces CMBPO, which integrates model-based reinforcement learning with constrained policy optimization. Learned dynamics models allow robots to imagine consequences before acting, dramatically reducing the real-world experience required for learning. However, imagined predictions contain errors, and mistakes about safety can lead to real harm. To address this challenge, CMBPO applies constrained trust-region optimization to model-generated rollouts, computing reward and cost advantages from synthetic trajectories while preserving the theoretical guarantees of CPO. Two uncertainty management mechanisms make this possible: adaptive mixing adjusts the balance between real and synthetic data based on ensemble disagreement, and dynamic rollout horizons terminate trajectories when cumulative uncertainty exceeds acceptable thresholds.

Experiments demonstrated 10 to 20 times reductions in sample complexity compared to model-free safe reinforcement learning methods while maintaining comparable asymptotic constraint satisfaction. Cumulative training costs remained significantly lower than the baselines, despite occasional transient violations during early learning.

7.1.2 Robotic Control Using Model-Based Meta-Adaptation

How can a robot adapt to novel conditions like damaged limbs or unfamiliar terrain without losing the movement patterns that matter?

Standard model-based meta-reinforcement learning adapts dynamics models to new conditions, achieving accurate predictions under novel circumstances. However, this creates a secondary problem: reward functions typically require manual re-engineering to achieve motion styles appropriate for each situation. A robot that correctly predicts its dynamics under damaged legs still needs carefully designed penalty terms to prevent dragging or hopping behaviors that maximize velocity but violate operational requirements. Chapter 4 addresses this through MAC, which uses reference task embeddings to encode preferred movement patterns and eliminates the need for manual reward tuning.

Dual-task conditioned dynamics models operate during deployment: one adapted to current conditions for accurate prediction, and one conditioned on a reference embedding corresponding to validated behaviors. A similarity-constrained model predictive control formulation maintains proximity between adapted and reference trajectories while maximizing task reward. Experimental validation across gravitational variations, morphological changes, and terrain modifications confirmed that MAC preserves stable gaits even when alternative behaviors would yield higher rewards.

7.1.3 Constrained Model-Agnostic Meta Learning

Can safety be transferred across tasks, so that a robot adapting to a new situation already knows how to stay safe?

Standard meta-reinforcement learning typically prioritizes rapid adaptation, optimizing for fast performance gains without explicitly accounting for constraint satisfaction. Chapter 5 introduces CMAML, which incorporates safety into both phases of meta-learning's structure: task-specific adaptation and meta-parameter optimization. This dual-level constraint enforcement ensures the meta-learned initialization not only adapts quickly but also systematically favors safe behaviors.

A central contribution is the meta safety critic, which observes constraint violations across tasks and steers meta-learning toward inherently safer initializations, using first-order gradient approximations to keep computation tractable. Experiments showed that CMAML produces initializations enabling rapid adaptation while maintaining costs near constraint thresholds, distinguishing it from standard meta-learning methods that adapt quickly but consistently violate safety limits. Validation with both TRPOLag and CPO as inner-loop optimizers confirmed the algorithm-agnostic design.

7.1.4 Scalable Off-Policy Learning for Vision-Based Locomotion

Can a robot efficiently learn vision-based walking in simulation and transfer this skill to real terrain?

Chapter 6 tackles the practical deployment challenges that arise when transitioning learned policies from simulation to physical hardware. While previous chapters established theoretical foundations for safe and adaptive learning, this work confronts the computational and perceptual obstacles that limit real-world applicability. Learning to walk from vision is difficult because the robot must simultaneously infer terrain geometry and decide how to move. Addressing this requires architectural and algorithmic choices that separate perception from control while maintaining sample efficiency under the memory constraints of visual processing.

An asymmetric actor-critic architecture separates these problems: critics access ground-truth terrain during training to provide clear learning signals, while policies learn using only deployable sensors. Off-policy learning enables reuse of past visual experiences rather than discarding them after each update, dramatically improving sample efficiency despite the computational demands of processing depth images. Recurrent temporal fusion extracts latent terrain properties from ego-centric depth sequences, and a hybrid training objective combines reinforcement learning with distillation from privileged teachers. Deployment on the physical Unitree GO2 confirmed successful sim-to-real transfer, with the robot navigating real obstacles using only its onboard depth camera.

7.2 Relationships Between Contributions

The four technical chapters address distinct aspects of safe, efficient, and adaptive robotic learning, but they share conceptual foundations and offer complementary solutions that could be integrated in future systems.

7.2.1 Safety Across Learning Paradigms

Chapters 3 and 5 both address safety, but at different levels of the learning hierarchy. Within individual tasks, CMBPO ensures constraint satisfaction by managing model uncertainty during policy optimization. Across task distributions, CMAML produces initializations that facilitate safe behavior in diverse scenarios encountered during deployment. Together, these contributions suggest a unified architecture: meta-training with CMAML could produce safety-aware initializations that are subsequently fine-tuned using CMBPO’s uncertainty-managed model-based learning.

Explicit constraints and behavioral appropriateness also warrant attention. CMBPO handles safety through cost functions and threshold satisfaction, while MAC enforces behavioral preferences through trajectory similarity constraints. These mechanisms target different aspects of operational acceptability. Cost constraints prevent specific undesirable outcomes such as collisions or joint limit violations; behavioral constraints maintain qualitative properties such as gait patterns or movement smoothness that resist specification through scalar cost functions. A comprehensive safety system might integrate both: cost constraints for quantifiable hazards and behavioral constraints for qualitative operational requirements.

7.2.2 Efficiency Through Imagination and Memory

Sample efficiency in this dissertation emerges from two complementary mechanisms. Chapters 3 and 4 achieve efficiency through imagination: learned dynamics models generate synthetic experience, allowing policies or planners to practice without additional physical interaction. Chapter 6 pursues a different approach through memory-based learning: off-policy methods store past experience in replay buffers and reuse it for many gradient updates instead of discarding it after each iteration.

These strategies target different bottlenecks. Model-based imagination reduces the need for real-world data when interaction is expensive or safety-critical. Replay-based memory amortizes the cost of data collection across large batches and many updates in simulation. Combining both approaches offers a promising direction: model-generated imagination could enrich replay buffers, further scaling data-efficient learning from limited but carefully collected real-world data.

7.2.3 Adaptation Mechanisms Across Timescales

Adaptation in this dissertation assumes fully observable Markovian states, where the robot can reliably infer its current operating conditions and adjust behavior accordingly. MAC adapts dynamics models through embedding optimization, enabling rapid adjustment to changes such as damaged limbs or modified terrain. CMAML adapts policy parameters through gradient-based fine-tuning, producing task-specific behaviors from a meta-learned initialization.

These methods operate at different timescales. Embedding optimization in MAC requires only a few observations to update task representations, making it suitable for fast responses to abrupt changes. Gradient-based adaptation in CMAML requires more experience but can modify policy structure more deeply, enabling adaptation to substantially different tasks.

Chapter 6 pursues a different route entirely. Rather than adapting online to current conditions, domain randomization expands the source domain to overlap with the target domain, training policies that are robust to variation without explicitly

identifying the underlying condition. This yields policies that function across diverse terrains but accept a loss in optimality for any specific one.

7.3 Future Directions

The methods developed in this dissertation open several promising research directions that extend their capabilities and address current limitations.

7.3.1 Scalability to Complex Task Distributions

Experimental validations in this dissertation focused on locomotion tasks with relatively limited task variation. Scaling to broader task distributions introduces several challenges. A single reference embedding in MAC cannot capture context-dependent behavioral preferences when different situations demand fundamentally different movement patterns. Meta-learning in CMAML becomes computationally intensive as task diversity increases, and the assumption of shared constraint structures across tasks may not hold in realistic multi-task scenarios.

Addressing these limitations requires extending both methods to handle greater diversity. For MAC, learning multiple reference embeddings could capture appropriate behaviors for different contexts. For CMAML, task-conditioned safety critics could enable handling heterogeneous constraint structures across diverse tasks. How to incorporate these structures while maintaining the safety and efficiency properties established in this dissertation remains an important open question.

7.3.2 Real-World Deployment and Continual Learning

Chapter 6 demonstrated sim-to-real transfer for vision-based locomotion, yet substantial gaps remain between these results and truly autonomous deployment. Real robots must continually adapt to changing conditions while maintaining safety, incorporate new experience into their models without catastrophic forgetting, and handle distribution shifts that exceed the variation encountered during training.

Extending CMBPO’s uncertainty estimates to detect when deployment conditions exceed reliable operating bounds could enable safe degradation: robots would recognize their limits and request human assistance rather than attempting dangerous behaviors. Meanwhile, MAC’s behavioral constraints could serve as anchors that prevent gradual drift toward inappropriate behaviors during prolonged deployment. Integrating these mechanisms with continual learning systems that update models and policies from ongoing experience remains essential for practical long-term autonomy.

7.4 Concluding Remarks

This dissertation has developed algorithmic foundations for learning robotic control policies that are simultaneously safe, efficient, and adaptive. From CMBPO's constrained model-based optimization through MAC's behaviorally-constrained adaptation and CMAML's safety-aware meta-training to Chapter 6's practical deployment system, a coherent vision emerges: robots that learn from limited experience while respecting operational constraints and adapting to the conditions they encounter.

Experimental validation across simulated locomotion tasks and physical robot deployment demonstrates feasibility. Substantial challenges remain in scaling these techniques to unstructured environments, heterogeneous task distributions, and long-horizon deployment scenarios where robots must continually learn and adapt. The methods presented here provide foundational components; realizing their full potential in autonomous systems that operate reliably over extended periods remains the work ahead.

Appendix

A Appendix to Chapter 3

A.1 Proof of Theorem 3.1: Return difference under model uncertainty

This appendix provides a proof sketch for Theorem 3.1 (stated in Sec. 3.4) using the performance-difference lemma (Kakade and Langford 2002) and standard total-variation (TV) perturbation bounds as used in TRPO analyses (Schulman et al. 2015).

Preliminaries and notation

Let P denote the true dynamics and P_m a learned model. For a policy π , let d^π and d_m^π be the stationary discounted state distributions under P and P_m , respectively. Define the model-based value and advantage by

$$V_m^\pi(s) = \mathbb{E}_{\tau \sim (\pi, P_m)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \mid s_0 = s \right],$$

$$A_m^\pi(s, a) = \mathbb{E}_{s' \sim P_m(\cdot | s, a)} [r(s, a, s') + \gamma V_m^\pi(s')] - V_m^\pi(s).$$

Abbreviate the TV distances

$$\epsilon_\pi := \max_s D_{\text{TV}}(\pi'(\cdot | s) \parallel \pi(\cdot | s)), \quad \epsilon_m := \max_{s, a} D_{\text{TV}}(P(\cdot | s, a) \parallel P_m(\cdot | s, a)).$$

Let the span proxy

$$\delta_{\max} := \max_{s, a, s'} \left| r(s, a, s') + V_m^\pi(s') - V_m^\pi(s) \right|$$

bound one-step temporal differences under V_m^π . Finally, define the model-relative performance term

$$L_m^\pi(\pi') := \mathbb{E}_{\substack{s \sim d_m^\pi \\ a \sim \pi'}} \left[\frac{\pi'(a | s)}{\pi(a | s)} A_m^\pi(s, a) \right].$$

Step 1: Performance-difference lemma under true dynamics

By the performance-difference lemma (Kakade and Langford 2002),

$$\Delta J(\pi, \pi') = J(\pi') - J(\pi) = \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^{\pi'}, a \sim \pi'} [A^\pi(s, a)]. \quad (\text{A.1})$$

Step 2: Replace A^π by A_m^π (model mismatch at one step)

For any fixed (s, a) , write

$$A^\pi(s, a) - A_m^\pi(s, a) = \mathbb{E}_{s' \sim P(\cdot|s, a)}[r + \gamma V_m^\pi(s')] - \mathbb{E}_{s' \sim P_m(\cdot|s, a)}[r + \gamma V_m^\pi(s')].$$

By the TV inequality $|\mathbb{E}_p[f] - \mathbb{E}_q[f]| \leq \text{span}(f) D_{\text{TV}}(p||q)$ and the definition of $\delta_{\max}$,

$$|A^\pi(s, a) - A_m^\pi(s, a)| \leq 2 \delta_{\max} \epsilon_m. \quad (\text{A.2})$$

Step 3: Move expectations from $(d^{\pi'}, \pi')$ to (d^π, π') to (d_m^π, π)

Using standard occupancy perturbation bounds (see, e.g., Lemma 1–2 style arguments in Schulman et al. 2015), policy shifts satisfy

$$\left| \mathbb{E}_{s \sim d^{\pi'}, a \sim \pi'}[f(s, a)] - \mathbb{E}_{s \sim d^\pi, a \sim \pi'}[f(s, a)] \right| \leq \frac{2\gamma}{1-\gamma} \epsilon_\pi \|f\|_\infty. \quad (\text{A.3})$$

Similarly, model vs. true dynamics under a fixed policy satisfy an analogous occupancy bound

$$\left| \mathbb{E}_{s \sim d^\pi, a \sim \pi'}[f(s, a)] - \mathbb{E}_{s \sim d_m^\pi, a \sim \pi'}[f(s, a)] \right| \leq \frac{2\gamma}{1-\gamma} \epsilon_m \|f\|_\infty, \quad (\text{A.4})$$

obtained by viewing the discounted visitation recursion as a γ -contraction and perturbing the transition kernel in TV (cf. MBPO-style analyses of model error terms (Janner et al. 2019)). Taking $f = A_m^\pi$ and using $|A_m^\pi| \leq 2\delta_{\max}$ (by the definition of $\delta_{\max}$) yields

$$\left| \mathbb{E}_{d^{\pi'}, \pi'}[A_m^\pi] - \mathbb{E}_{d_m^\pi, \pi'}[A_m^\pi] \right| \leq \frac{4\gamma}{1-\gamma} \delta_{\max} (\epsilon_\pi + \epsilon_m). \quad (\text{A.5})$$

Step 4: Replace π' by importance ratios under π

Express the remaining policy change by importance weights:

$$\mathbb{E}_{s \sim d_m^\pi, a \sim \pi'}[A_m^\pi(s, a)] = \mathbb{E}_{s \sim d_m^\pi, a \sim \pi} \left[\frac{\pi'(a|s)}{\pi(a|s)} A_m^\pi(s, a) \right] + R_{\text{IS}},$$

where the remainder R_{IS} is controlled by TV between π' and π . Using $\left| \frac{\pi'}{\pi} - 1 \right| \leq 2 D_{\text{TV}}(\pi'||\pi)[s]$ pointwise and $|A_m^\pi| \leq 2\delta_{\max}$,

$$|R_{\text{IS}}| \leq 4 \delta_{\max} \epsilon_\pi. \quad (\text{A.6})$$

Combining yields

$$\mathbb{E}_{s \sim d_m^\pi, a \sim \pi'}[A_m^\pi(s, a)] = L_m^\pi(\pi') \pm 4 \delta_{\max} \epsilon_\pi. \quad (\text{A.7})$$

Step 5: Put the pieces together

Start from (A.1), substitute $A^\pi = A_m^\pi \pm 2\delta_{\max}\epsilon_m$ from (A.2), then apply the expectation shifts (A.5) and the importance relation (A.7). Collecting terms and simplifying gives

$$\Delta J(\pi, \pi') = \frac{1}{1-\gamma} \mathbb{E}_{d_m^\pi, \pi} \left[\frac{\pi'}{\pi} A_m^\pi \right] \pm \frac{4\delta_{\max}}{1-\gamma} \left(\epsilon_\pi \epsilon_m + \frac{\gamma}{1-\gamma} (\epsilon_\pi^2 + 2\epsilon_\pi \epsilon_m) \right),$$

which matches Theorem 3.1 with

$$\epsilon = \epsilon_\pi \epsilon_m + \frac{\gamma}{1-\gamma} (\epsilon_\pi^2 + 2\epsilon_\pi \epsilon_m).$$

□

Remarks. (i) Constants depend only on TV/KL conventions and span bounds; using Pinsker's inequality ($D_{\text{TV}}^2 \leq \frac{1}{2} D_{\text{KL}}$) recovers familiar KL forms as in Schulman et al. (2015). (ii) The structure mirrors MBPO's model-usage analysis, which also decomposes error into policy divergence and model error terms (Janner et al. 2019, Theorem 4.1).

A.2 CPO step details

This appendix gives the compact derivation of the CPO update used in Sec. 3.4.2. We reuse the same notation: current parameters θ , proposal θ' , step $x := \theta' - \theta$, objective gradient $g = \nabla_\theta L_m^\pi(\pi_\theta)$, cost gradient $b = \nabla_\theta L_{m,c}^\pi(\pi_\theta)$, Fisher matrix H , KL radius d_π , and cost slack $c_0 := J_c(\pi_\theta) - d_c$. We also define the scaled cost-gradient

$$\tilde{b} := \frac{b}{1-\gamma}. \quad (\text{A.8})$$

Lagrangian and stationarity. Under the local approximations (Eqs. 3.16–3.18), the Lagrangian of the subproblem (Eq. 3.19) is

$$\mathcal{L}(x, \lambda, \nu) = g^\top x - \nu \left[\frac{1}{2} x^\top H x - d_\pi \right] - \lambda \left[c_0 + \tilde{b}^\top x \right]. \quad (\text{A.9})$$

Taking the gradient with respect to x and setting to zero gives

$$g - \nu H x - \lambda \tilde{b} = 0 \implies x = \frac{1}{\nu} H^{-1} (g - \lambda \tilde{b}). \quad (\text{A.10})$$

Dual scalars and useful abbreviations. Introduce

$$A := g^\top H^{-1} g, \quad r := g^\top H^{-1} \tilde{b}, \quad S := \tilde{b}^\top H^{-1} \tilde{b}. \quad (\text{A.11})$$

With (A.10), the two constraints (active at the solution when binding) yield

$$\frac{1}{2}x^\top Hx = d_\pi \implies \nu = \sqrt{\frac{A - 2\lambda r + \lambda^2 S}{2d_\pi}}, \quad (\text{A.12})$$

$$c_0 + \tilde{b}^\top x = 0 \implies \nu = \frac{r - \lambda S}{-c_0}. \quad (\text{A.13})$$

Equating (A.12) and (A.13) gives a quadratic for λ :

$$\underbrace{(c_0^2 S - 2d_\pi S^2)}_{=:a} \lambda^2 + \underbrace{(-2c_0^2 r + 4d_\pi r S)}_{=:b} \lambda + \underbrace{(c_0^2 A - 2d_\pi r^2)}_{=:c} = 0. \quad (\text{A.14})$$

Choose the root λ^* that yields $\nu^* > 0$ via either (A.12) or (A.13). The step and updated parameters are then

$$x^* = \frac{1}{\nu^*} H^{-1}(g - \lambda^* \tilde{b}), \quad \theta^* = \theta + x^*. \quad (\text{A.15})$$

Inactive-cost fallback. If the unconstrained TRPO direction satisfies the linearized cost (i.e., with $\lambda=0$ and $x \propto H^{-1}g$ we have $c_0 + \tilde{b}^\top x \leq 0$), then the cost constraint is inactive. In that case, set $\lambda^* = 0$ and determine ν^* from the KL equality in (A.12).

Practical notes. We compute $H^{-1}v$ products by conjugate gradients; H is the Fisher (KL Hessian), so Hessian–vector products are estimated from policy samples. We use a backtracking line search on (A.15) to ensure the *empirical* KL and cost constraints hold. In CMBPO, model uncertainty is enforced outside this step (adaptive resampling and uncertainty-capped rollouts), so no additional multiplier appears here.

A.3 Hyperparameters

Tables A.1 and A.2 report all hyperparameters used in the CMBPO experiments. Common architectural and algorithmic choices shared across all environments are listed in Table A.1, while environment-specific settings obtained through per-task tuning are provided in Table A.2.

Hyperparameter	Value
<i>Policy Network</i>	
Hidden layers	2×256
Activation	ReLU
<i>Value Network</i>	
Hidden layers	2×128
Activation	ReLU
Learning rate	3×10^{-4}
Batch size	2048
Training repeats	10
<i>Trust Region Optimization</i>	
Conjugate gradient steps	10
Conjugate gradient damping	0.1
Backtracking steps	10
GAE discount λ	0.95
Discount factor γ (reward)	0.99
<i>Dynamics Model</i>	
Hidden layers	2×512
Ensemble size	7
Ensemble elites	5
Learning rate	1×10^{-3}
Training batch size	2048
Rollout termination ratio δ_r	3.5

Table A.1: Hyperparameters shared across all environments. The policy uses a trust region update with conjugate gradient optimization; the value function is trained by regression on GAE targets. The dynamics model is a probabilistic ensemble with adaptive rollout termination.

Hyperparameter	Ant	HC	AntSafe	HCSafe	PointGoal2
<i>Constrained Optimization</i>					
Cost limit	–	–	10	10	25
Discount γ (cost)	–	–	0.97	0.97	0.97
<i>Policy Learning</i>					
Target KL	0.04	0.02	0.015	0.02	0.02
Policy batch size	8 092	6 000	20 000	25 000	50 000
Entropy regularization	10^{-4}	2×10^{-3}	10^{-4}	2×10^{-3}	2×10^{-3}
<i>Dynamics Model</i>					
Loss	MSE	MSE	MSE	MSE	NLL
Start state batch size	300	200	200	200	1 000
Max. model error $\epsilon_m^{\max}$	0.065	0.075	0.05	0.03	0.1
Epoch sampling discount d^γ	0.995	0.995	0.995	0.995	0.95

Table A.2: Environment-specific hyperparameters for CMBPO. HC denotes HalfCheetah. Constrained environments (AntSafe, HCSafe, PointGoal2) include cost discounting and cost limits; unconstrained environments (Ant, HC) omit these. All values were obtained through per-environment tuning.

B Appendix to Chapter 4

B.1 Hyperparameters

Tables B.1–B.3 report all hyperparameters used in the experiments. Common architectural choices are shared across methods (Table B.1), while meta-training (Table B.2) and meta-testing (Table B.3) hyperparameters were obtained via environment-specific hyperparameter sweeps using Weights & Biases.

Hyperparameter	Value
<i>Network Architecture</i>	
Hidden layers	3
Hidden units per layer	512
<i>Meta-Training (shared)</i>	
Meta iterations	100 000
Meta batch size	32
Learning rate schedule	Cosine annealing
Train/test split	0.8
Inner sample size	500
Random seed	42
<i>Online Adaptation (shared)</i>	
Episode steps	500
Action bounds	$[-1, 1]$
Adaptation basis	Dynamics
<i>MAC-Specific (shared)</i>	
MAC elite count	512
CEM elite fraction	0.1
CEM smoothing α	0.1
Population size	8 192
Discount factor	1.0
CEM enabled	Yes

Table B.1: Hyperparameters shared across all environments and methods. Network architecture is identical for all dynamics models. MAC-specific parameters apply to the MAC controller; FMPC and RMPC use their own controller configurations listed in Table B.3.

Hyperparameter	FAMLE				REPTILE			
	<i>Ant-Grav.</i>	<i>Ant-Dis.</i>	<i>HC-Dis.</i>	<i>HC-Pier</i>	<i>Ant-Grav.</i>	<i>Ant-Dis.</i>	<i>HC-Dis.</i>	<i>HC-Pier</i>
Activation	SiLU	ReLU	Leaky	ReLU	SiLU	SiLU	ReLU	Leaky
Embedding dim	2	8	10	10	–	6	4	4
Dropout	0.05	0	0	0.075	0.025	0	0.15	0
Meta step α_{meta}	0.9	0.5	0.8	0.2	1.0	1.1	1.0	1.2
Inner iterations	197	193	194	162	50	50	47	49
Inner step α_{inner}	0.001	0.003	0.002	0.003	0.003	0.0008	0.003	0.0006
Training rollouts	100	600	600	400	100	600	600	400
Episode length	1000	500	500	500	1000	500	500	500
Training tasks	17	6	6	11	17	6	6	11

Table B.2: Meta-training hyperparameters per algorithm and environment. All methods share the same network architecture (3×512) and meta batch size (32). FAMLE uses task embeddings for efficient adaptation, while REPTILE adapts full model parameters. Environment-specific values were obtained through hyperparameter sweeps.

Param.	MAC				FMPC				RMPC			
	<i>Ant-G.</i>	<i>Ant-D.</i>	<i>HC-D.</i>	<i>HC-P.</i>	<i>Ant-G.</i>	<i>Ant-D.</i>	<i>HC-D.</i>	<i>HC-P.</i>	<i>Ant-G.</i>	<i>Ant-D.</i>	<i>HC-D.</i>	<i>HC-P.</i>
<i>Online Model Adaptation</i>												
β	2.1	1.6	0.9	1.8	0.6	2.5	0.3	0.3	–	–	–	–
Epochs	165	175	70	76	75	66	80	137	108	50	198	78
LR	4e-3	7e-4	4e-6	6e-4	3e-5	6e-5	5e-4	8e-6	1e-4	8e-3	4e-4	3e-3
m obs.	23	13	16	28	25	29	25	12	28	19	14	11
Minibatch	32	32	32	32	32	32	32	32	11	11	11	11
Iterations	8	1	2	2	2	1	2	2	2	1	2	2
<i>Planning / Controller</i>												
Horizon	17	10	23	20	11	13	23	12	26	16	22	21
Popsiz	8192	8192	8192	8192	4096	4096	4096	8192	4096	4096	4096	4096
Discount	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.99	.99	.99	.99
CEM	✓	✓	✓	✓	–	–	–	–	–	–	–	–

Table B.3: Meta-testing hyperparameters per method and environment. Ant-G. = Ant-Gravity, Ant-D. = Ant-Disabled, HC-D. = HalfCheetah-Disabled, HC-P. = HalfCheetah-Pier. β controls the FAMLE embedding adaptation rate. m obs. denotes the number of initial exploration observations before adaptation. LR is the online adaptation learning rate. MAC uses CEM action sampling with 512 elite trajectories; FMPC and RMPC use uniform random sampling. All environment-specific values were obtained through hyperparameter sweeps.

C Appendix to Chapter 5

C.1 Hyperparameters

Tables C.1–C.2 report all hyperparameters used in the CMAML experiments. Shared architecture and algorithm settings are listed in Table C.1. Meta-learning and fine-tuning parameters are given in Table C.2. All methods share identical network architectures and training configurations to ensure fair comparison.

Hyperparameter	Value
<i>Network Architecture</i>	
Actor hidden layers	[64, 64]
Critic hidden layers	[64, 64, 64]
Activation	tanh
Weight initialization	Kaiming uniform
Actor type	Gaussian (learned std)
Std range	[0.5, 0.1]
<i>Inner Loop Algorithm (shared across CPO / TRPOLag / TRPO)</i>	
Steps per epoch	20 000
Critic update iterations	10
Critic learning rate	0.01
Batch size	128
Discount factor (reward) γ	0.999999
Discount factor (cost) γ_c	0.999999
GAE λ (reward)	0.95
GAE λ (cost)	0.95
Advantage estimation	plain
Standardized advantages	disabled
Max gradient norm	40.0
Entropy coefficient	0.0
Linear LR decay	disabled
<i>Trust Region</i>	
Target KL	0.01
CG damping	0.1
CG iterations	15

Table C.1: Architecture and inner loop algorithm hyperparameters shared across all methods. Network architecture is identical for all baselines. The actor learning rate is not set explicitly, as TRPO and CPO compute policy updates via conjugate gradient optimization subject to the KL constraint.

Hyperparameter	Value
<i>Meta-Training (FOMAML Outer Loop)</i>	
Outer loop iterations	100
Tasks per meta-batch	20
Inner loop adaptation steps	3
Outer loop actor optimizer	SGD (lr = 1.0)
Outer loop cost critic optimizer	Adam (lr = 10^{-3})
Number of navigation tasks	106 (Env1) / 107 (Env2)
<i>C-MAML Safety Mechanism</i>	
Meta Lagrange multiplier η init	0.01
η learning rate	0.035
<i>TRPOLag Inner Loop</i>	
Cost limit d	5.0
Lagrange multiplier init	0.01
λ learning rate	0.035
λ optimizer	Adam
<i>CPO Inner Loop</i>	
Cost limit d	10.0
<i>Fine-Tuning</i>	
Fine-tuning epochs	20
Steps per fine-tuning epoch	20 000
Evaluation seeds	7

Table C.2: Meta-learning outer loop, safety mechanism, and fine-tuning hyperparameters. The outer loop uses first-order MAML (FOMAML) with SGD for actor meta-updates. The meta Lagrange multiplier η regulates the influence of the meta safety critic during outer loop optimization. The cost limit differs between TRPOLag ($d = 5$) and CPO ($d = 10$) to account for differences in constraint handling between the two algorithms.

D Appendix to Chapter 6

D.1 Visual Augmentation and Normalization

We apply targeted visual augmentation during training to regularize the depth encoder and improve robustness to viewpoint jitter, sensor noise, and partial occlusions. Each depth frame is transformed in sequence:

- **Replicate-padded random shift:** replicate-pad by 4 pixels on all sides, then crop a random 48×48 window (equivalent to integer shifts in $[-4, 4]$ along x and y)
- **Additive Gaussian noise:** add i.i.d. noise $\mathcal{N}(0, 0.02^2)$, applied to 50% of the images
- **Random cutout:** sample one rectangle per image with height and width uniformly drawn from $[2, 24]$, place it at a random location, and fill it with either zeros, ones, or i.i.d. uniform $[0, 1]$ values; a fill is applied with probability 0.6

Proprioceptive and visual observations are normalized online using running statistics. For each observation dimension x we compute

$$\hat{x} = \frac{x - \mu_{\text{running}}}{\sqrt{\sigma_{\text{running}}^2 + \epsilon}}, \quad \epsilon = 10^{-2}, \quad (\text{D.1})$$

where μ_{running} and $\sigma_{\text{running}}^2$ are exponential moving averages updated during data collection. This normalization keeps input magnitudes stable across parallel environments and observation modalities.

D.2 Recurrent Off-Policy Sequence Training

Training recurrent policies from replay buffers requires modifications to standard off-policy procedures. The replay buffer stores transitions augmented with recurrent hidden states and samples fixed-length sequences rather than individual transitions.

Sequence Sampling. Each training batch consists of contiguous trajectory segments of fixed length L . For a sampled start index t , the sampler constructs a sequence of consecutive transitions

$$\left\{ (s_{t+k}, a_{t+k}, r_{t+k}, s_{t+k+1}, d_{t+k}) \right\}_{k=0}^{L-1},$$

where d_{t+k} is the terminal flag. This produces batches with shape $[B/L, L, \cdot]$ where B is the batch size and L is the sequence length, maintaining temporal structure within each segment.

Terminal Masking. Sequences that cross episode boundaries require special handling to prevent bootstrapping across unrelated trajectories. A binary mask is computed from the terminal flags within each sequence, taking value one before the first terminal and zero afterwards. This mask multiplies both the loss terms and the multi-step return targets, ensuring that gradients do not propagate through episode boundaries and that bootstrap targets use only valid future rewards.

Hidden State Storage. When enabled, the buffer records the recurrent hidden state associated with each transition alongside observations and actions. For GRU networks this corresponds to the hidden vector; for LSTMs it includes both hidden and cell states. Storing these states allows replayed sequences to initialize from the original hidden context rather than resetting to zero, preserving the temporal information available during data collection.

Together these modifications maintain consistency between the hidden state evolution during data collection and the recurrent updates during training, enabling stable off-policy learning with temporal credit assignment.

D.3 Hyperparameters

Tables D.1–D.3 report all hyperparameters used in the experiments. Shared architectural and algorithmic choices are listed in Table D.1. Vision and recurrent settings that apply only to the depth-based student are given in Table D.2. Hybrid training parameters that govern the knowledge distillation weight and teacher integration are listed in Table D.3.

Hyperparameter	Value
<i>Environment</i>	
Parallel environments	4 096 (proprioceptive) / 512 (vision)
Evaluation environments	128
<i>Training</i>	
Discount factor γ	0.99
Target smoothing τ	0.1
Multi-step returns	1
Updates per step	2
Delayed policy updates	2
Learning starts	10 steps
Automatic mixed precision	bf16
Torch compile mode	reduce-overhead
<i>Optimization</i>	
Critic learning rate	3×10^{-4}
Actor learning rate	3×10^{-4}
Weight decay	0.1
Gradient norm clipping	disabled
<i>Replay Buffer</i>	
Buffer size	51 200
Batch size	32 768
<i>Exploration Noise</i>	
Noise scheduling	enabled
$\sigma_{\min}$	0.001
$\sigma_{\max}$	0.4
Target policy noise	0.001
Noise clip	0.5
<i>Action Space</i>	
Squashing (tanh)	enabled
Action bounds	$[-1, 1]$
<i>Distributional Critic (C51)</i>	
Number of atoms	251
$V_{\min}$	-250.0
$V_{\max}$	250.0
Clipped double Q-learning	enabled

Table D.1: Hyperparameters shared across all methods. The number of parallel environments differs between proprioceptive and vision-based training due to GPU memory constraints imposed by the depth encoder. All other settings are identical.

Hyperparameter	Value
<i>Network Architecture (SimbaV2)</i>	
Critic hidden dimension	1 024
Actor hidden dimension	512
Critic residual blocks	2
Actor residual blocks	1
Initialization scale	0.01
Layer normalization	disabled
<i>Vision Encoder</i>	
Depth input resolution	48×48
Vision latent dimension	128
Vision update rate	1
Vision layer normalization	disabled
Concatenate proprioception	enabled
Use next-step vision observation	enabled
<i>Recurrent Module (GRU)</i>	
Memory type	GRU
Hidden dimension	256
Sequence length	50
Burn-in length	10
Store hidden state in buffer	disabled
Sequential critic	disabled
<i>Visual Augmentation</i>	
Random spatial shift	± 4 pixels (replicate pad)
Additive Gaussian noise	$\mathcal{N}(0, 0.02^2)$, applied to 50% of frames
Random cutout	rectangle $[2, 24]$ pixels, probability 0.6
<i>Normalization</i>	
Observation normalization	enabled (running statistics)
Reward normalization	disabled
Normalization ϵ	10^{-2}

Table D.2: Network architecture, vision encoder, and recurrent module hyperparameters. The SimbaV2 architecture uses hyperspherical normalization for both actor and critic. The vision encoder is kept deliberately lightweight to preserve GPU memory for parallel environments. Visual augmentation is applied only during training.

Hyperparameter	Value
<i>Hybrid Training</i>	
Hybrid mode	enabled
Initial distillation weight $\lambda_{\max}$	1.0 (default)
Decay rate α	3.0
<i>Knowledge Distillation</i>	
BC coefficient α_{bc}	0.1
Offline mode	disabled
<i>RLPD</i>	
Teacher buffer size	2 048 000 transitions
Online/offline sampling ratio	50% / 50%

Table D.3: Hyperparameters specific to the teacher-student training variants. For the hybrid approach, $\lambda_{\max}$ controls the initial strength of the knowledge distillation term and α governs its exponential decay. Section 6.5 reports ablations over $\lambda_{\max} \in \{0.01, 0.1, 0.5, 1.0\}$ and $\alpha \in \{0, 3\}$. The RLPD variant samples each mini-batch with equal proportions from the fixed teacher buffer and the student’s growing replay buffer.

List of Figures

2.1	Markov Decision Process Framework	10
2.2	Actor-Critic Architecture	12
2.3	Constrained Actor-Critic Architecture	16
2.4	Model-Based Reinforcement Learning Framework	17
2.5	Meta-Learning Framework	19
3.1	Ensemble Uncertainty in Learned Dynamics Models	27
3.2	Ensemble Predictions and KL-Based Disagreement	32
3.3	Evaluation Environments for Constrained Control Tasks	35
3.4	Training Performance Across Algorithms and Environments	37
3.5	Ablation Study on AntSafe and AntCircle	39
4.1	MAC Architecture for Meta-Training and Online Adaptation	50
4.2	Evaluation Environments for MAC Experiments	60
4.3	Behavioral Comparison on Ant Gravity Tasks	62
5.1	Constrained Meta-Learning in Policy Parameter Space	72
5.2	SafetyGymnasium Navigation Task Distribution and Point Robot	77
5.3	Effect of the Meta Safety Critic	80
5.4	Fine-Tuning Performance With TRPOLag	81
5.5	Fine-Tuning Performance With CPO	82
6.1	Hybrid Teacher-Student Training Architecture	94
6.2	Effect of Action Scaling on Leg Joint Commands	95
6.3	Simulation Terrains for Locomotion Training	97
6.4	Deployment System Architecture for Real-World Locomotion	98
6.5	Terrain-Level Performance Across Training	100
6.6	Mean Reward Comparison Across Hybrid Training Variants	101
6.7	Effect of Knowledge Distillation Weight on Learning	101
6.8	Geometric Setup of the Crawling Task	102
6.9	Real-World Crawling Behavior Under a Low Obstacle	103

List of Tables

1.1	Mapping of Dissertation Contributions to Core Research Themes	7
3.1	CMBPO: Total Constraint Violations	38
4.1	Task Environment Specifications	59
4.2	MAC: Meta-Testing Returns	61
A.1	Common CMBPO Hyperparameters	119
A.2	Environment-Specific CMBPO Hyperparameters	120
B.1	Common MAC Hyperparameters	122
B.2	MAC: Meta-Training Hyperparameters	123
B.3	MAC: Meta-Testing Hyperparameters	124
C.1	Shared C-MAML Hyperparameters	126
C.2	C-MAML Meta-Learning and Fine-Tuning Hyperparameters	127
D.1	Shared Hyperparameters for Off-Policy Locomotion	131
D.2	Architecture and Vision Hyperparameters	132
D.3	Hybrid and Teacher-Student Hyperparameters	133

Own Publications

This list contains all publications where the author of this dissertation is either the first author or has significantly contributed as a co-author (in terms of problem formulation, solution, discussion, or experimental evaluation).

Tobias Fleck, Karam Daaboul, Michael Weber, Philip Schörner, Marek Wehmer, Jens Doll, Stefan Orf, Nico Sussmann, Christian Hubschneider, Marc René Zofka, Florian Kuhnt, Ralf Kohlhaas, Ingmar Baumgart, Raoul Zöllner, and J. Marius Zöllner. Towards large scale urban traffic reference data: Smart infrastructure in the test area autonomous driving baden-württemberg. In *15th International Conference on Intelligent Autonomous Systems (IAS-15)*. Springer, 2018.

Karam Daaboul, Simon Becker, Kevin Daiss, Marcus Geimer, and J. Marius Zöllner. Machine learning for process automation of mobile machines in field applications. In *Land. Technik AgEng 2019: 77th International Conference on Agricultural Engineering, Hannover*, 2019.

Karl Kurzer, Philip Schörner, Alexander Albers, Hauke Thomsen, Karam Daaboul, and J. Marius Zöllner. Generalizing decision making for automated driving with an invariant environment representation using deep reinforcement learning. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 871–878, 2021. doi: 10.1109/IV48863.2021.9575382.

Moritz A. Zanger, Karam Daaboul, and J. Marius Zöllner. Safe continuous control with constrained model-based policy optimization. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3512–3519, September 2021. doi: 10.1109/IROS51168.2021.9635984.

Karam Daaboul, Joel Ikels, and J. Marius Zöllner. Robotic control using model based meta adaption. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9457–9463, May 2023. doi: 10.1109/ICRA48891.2023.10160425.

Karam Daaboul, Florian Kuhm, Tim Joseph, and J. Marius Zöllner. Constrained meta agnostic reinforcement learning. In *CoRL 2024 Workshop SAFE-ROL*, October 2024.

Student Work

This list contains student theses supervised by the author during the course of this doctoral research.

Moritz Akiya Zanger and Karam Daaboul. Uncertainty-aware constrained model-based policy optimization. Master thesis, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2020.

Moritz Nekolla and Karam Daaboul. Efficient deep reinforcement learning in combination of variational autoencoders and soft actor critic on robotic tasks, 2020.

Pascal Schindler and Karam Daaboul. Offline reinforcement learning in autonomous driving, 2021.

Jan Bieser and Karam Daaboul. Optimization of latent dynamics models with unsupervised representation learning. Master thesis, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2022.

Adrian Wiedemann and Karam Daaboul. Contrastive offline reinforcement learning for pixel to control, 2022.

Florian Krone and Karam Daaboul. Safe transfer learning for robotic environments. Master thesis, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2022.

Hung Vu and Karam Daaboul. Using generative adversarial networks to enhance contrastive model-based optimization. Master thesis, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2023.

Henri Paul Simon and Karam Daaboul. Evaluating offline pretrained representations for robotic control tasks. Master thesis, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2023.

Florian Kuhm and Karam Daaboul. Constrained model-agnostic meta learning. Master thesis, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2024.

Maximilian Schik and Karam Daaboul. Optimizing off-policy reinforcement learning for scalable, real-time locomotion tasks. Master thesis, KIT Karlsruher Institut für Technologie, Karlsruhe, Germany, 2025.

Bibliography

- Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, page 1, 2004. doi: 10.1145/1015330.1015430.
- Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 22–31. PMLR, 2017.
- Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning (CoRL)*, pages 403–415. PMLR, 2023.
- Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.
- Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Learning Representations (ICLR)*, 2023.
- Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 449–458. PMLR, 2017.
- Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 908–918, 2017.
- Stephen Boyd and Lieven Vandenbergh. *Convex Optimization*. Cambridge University Press, 2004.
- Eduardo F. Camacho and Carlos Alba Bordons. *Model Predictive Control*. Springer Science & Business Media, 2nd edition, 2013. doi: 10.1007/978-0-85729-398-5.
- Edoardo Cetin, Philip J. Ball, Stephen Roberts, and Oya Celiktutan. Stabilizing off-policy deep reinforcement learning from pixels. In *International Conference on Machine Learning (ICML)*, pages 2784–2810. PMLR, 2022.
- Elliot Chane-Sane, Joseph Amigo, Thomas Flayols, Ludovic Righetti, and Nicolas Mansard. SoloParkour: Constrained reinforcement learning for visual locomotion from privileged experience. In *Conference on Robot Learning (CoRL)*, 2024.
- Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 15084–15097, 2021a.

- Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double Q-learning: Learning fast without a model. In *International Conference on Learning Representations (ICLR)*, 2021b.
- Minjae Cho and Chuangchuang Sun. Constrained meta-reinforcement learning for adaptable safety guarantee with differentiable convex programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20975–20983, 2024.
- Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(167):1–51, 2018a.
- Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A Lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8092–8101, 2018b.
- Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 4754–4765, 2018.
- Gal Dalal, Balázs Szörényi, Gugan Thoppe, and Shie Mannor. Finite sample analyses for TD(0) with function approximation. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 465–472, 2011.
- 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 *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1126–1135. PMLR, 2017.
- Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pages 1587–1596. PMLR, 2018.
- Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.
- Cong Guan, Ruiqi Xue, Ziqian Zhang, Lihe Li, Yi-Chen Li, Lei Yuan, and Yang Yu. Cost-aware offline safe meta reinforcement learning with robust in-distribution online task adaptation. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 743–751, 2024.

- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1861–1870. PMLR, 2018.
- Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 2555–2565. PMLR, 2019.
- Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.
- Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 4565–4573, 2016.
- David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. ANYmal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88): eadi7566, 2024. doi: 10.1126/scirobotics.adi7566.
- Marcel Hussing, Claas Voelcker, Igor Gilitschenski, Amir-massoud Farahmand, and Eric Eaton. Dissecting deep RL with high update ratios: Combatting value divergence. In *Reinforcement Learning Conference (RLC)*, 2024.
- Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.
- Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe reinforcement learning using probabilistic shields. In *International Conference on Concurrency Theory (CONCUR)*, pages 3:1–3:16, 2020.
- Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Juntao Dai, and Yaodong Yang. SafetyGymnasium: A unified safe reinforcement learning benchmark. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, volume 36, 2023.
- Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 267–274. Morgan Kaufmann, 2002.
- Rituraj Kaushik, Timothée Anne, and Jean-Baptiste Mouret. Fast online adaptation in robotics through meta-learning embeddings of simulated priors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5269–5276. IEEE, 2020. doi: 10.1109/IROS45743.2020.9341462.
- Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOREL: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 21810–21823, 2020.

- Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6059–6066. IEEE, 2018.
- Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems (RSS)*, 2021. doi: 10.15607/RSS.2021.XVII.011.
- Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations (ICLR)*, 2018.
- Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 6402–6413, 2017.
- Hoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R. Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. SimBa: Simplicity bias for scaling up parameters in deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025a.
- Hoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. *arXiv preprint arXiv:2502.15280*, 2025b.
- Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi: 10.1126/scirobotics.abc5986.
- Zechu Li, Tao Chen, Zhang-Wei Hong, Anurag Ajay, and Pulkit Agrawal. Parallel q -learning: Scaling off-policy reinforcement learning under massively parallel simulation. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pages 19440–19459. PMLR, 2023.
- Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. In *Proceedings of the Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, volume 1, 2021.
- David Q. Mayne, James B. Rawlings, Christopher V. Rao, and Pierre O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000. doi: 10.1016/S0005-1098(99)00214-9.
- Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022. doi: 10.1126/scirobotics.abk2822.

- Mayank Mittal, Calvin Yu, Qinxu Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharmashan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015. doi: 10.1038/nature14236.
- Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566, 2018.
- Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based RL. In *International Conference on Learning Representations (ICLR)*, 2019.
- Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. In *Proceedings of the NeurIPS 2018 Workshop on Meta-Learning*, 2018.
- Evgenii Nikishin, Max Schwarzer, Pierluca D’Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 16828–16847. PMLR, 2022.
- Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 4026–4034, 2016.
- Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2778–2787, 2017.
- Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems (RSS)*, 2017. doi: 10.15607/RSS.2017.XIII.008.
- Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991. doi: 10.1162/neco.1991.3.1.88.
- Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 5331–5340. PMLR, 2019.
- James B. Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2nd edition, 2017.

- Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019.
- Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.
- Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999. doi: 10.1023/A:1010091220143.
- Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning (CoRL)*, pages 91–100. PMLR, 2022.
- Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. Diploma thesis, Technische Universität München, 1987.
- John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.
- John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 8583–8592. PMLR, 2020.
- Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. FastTD3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.
- Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations (ICLR)*, 2019.
- Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

- Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. Safe reinforcement learning via curriculum induction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 12151–12162, 2020.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5998–6008, 2017.
- Akifumi Wachi and Yanan Sui. Safe reinforcement learning in constrained Markov decision processes. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 9797–9806. PMLR, 2020.
- Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.
- Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017. doi: 10.1109/ICRA.2017.7989202.
- Siyuan Xu and Minghui Zhu. Online constrained meta-learning: Provable guarantees for generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 16683–16705, 2023.
- Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, volume 22, pages 1433–1438, 2008.
- Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A very good method for bayes-adaptive deep RL via meta-learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Erklärung zur Nutzung generativer KI

Ich habe generative KI-Sprachmodelle bei der Erstellung dieser Dissertation eingesetzt und dokumentiere dies gemäß der Handreichung der KIT-Fakultät für Wirtschaftswissenschaften (Januar 2025).

- **Tools:** ChatGPT (OpenAI), Claude (Anthropic)
- **Einsatzbereiche:**
 - Sprachliche Überarbeitung sämtlicher Kapitel (Rechtschreibung, Grammatik, Stil, Klarheit)
 - Formulierungsvorschläge für einzelne Textpassagen
 - Unterstützung bei der LaTeX-Formatierung
 - Unterstützung bei der Implementierung des Programmcodes in Kapitel 6 (Off-Policy Locomotion)
- **Nicht eingesetzt für:** Entwicklung der Forschungsmethodik, Herleitung mathematischer Ergebnisse, Durchführung oder Auswertung von Experimenten, Erstellung von Abbildungen sowie Programmcode der Algorithmen in Kapiteln 3–5
- **Umfang der Nachbearbeitung:** Sämtliche KI-Ausgaben wurden von mir kritisch geprüft, inhaltlich verifiziert und eigenständig überarbeitet. Die Überarbeitung umfasste regelmäßig substanzielle Änderungen an Struktur, Wortwahl und fachlicher Präzision.

Für Inhalt, Argumentation sowie die Richtigkeit aller Aussagen, Zitate und Literaturangaben trage ich die volle Verantwortung. Es wurden keine vertraulichen oder personenbezogenen Daten in die Tools eingegeben.